# موسوعة البرمجة بلغة ++C

# C++ encyclopedia Programming



**المؤلف:** مجلاد مشاري السبيعي
**البريد الإلكتروني:** magedxl@hotmail.com

**يمكنكم مراسلتي إذا كان هناك آي مشاكل متعلقة بعدم التوضيح أو نحوه مما يوجد في هذا الكتاب ...**

*<u>**هذا الكتاب مجاني للجميع**</u>

# إهداء

**هذا الكتاب إهداء إلى ارض العراق الحبيبة .. نعم العراق لقد أضعنا العراق ارض الفرات ودجلة ، فخر الإسلام والمسلمين بـغـداد ...**
**أضعناها .. أضعناها..**
**لكنك لست ببعيد وستعودين أرضا للخلافه الإسلامية..**
**قريبا .. قريبا**

**اليك ياعراق**

# مقدمة عامة

أعزائي..

تم وبحمد الله الانتهاء من هذا الكتاب .. المفيد للجميع بإذن الله المبتدئين والمحترفين على حد السواء..
وقد قمت بإنشاء هذا الكتاب بعد أن لاحظت قلت الكتب التي تتحدث عن لغة Visual C++..
وهي اللغة التي تم منها بناء حزمة اوفيس المعروفة لتطبيقات الحاسب..
لذلك أرجو أن يحوز هذا الكتاب على رضاكم وهو يحتوى على العديد من الأمثلة التطبيقية مع شرح الأساسيات
والعديد من الأجوبة التي يحتاجها مبرمجي C++.
وبالله التوفيق..

# فهرس المحتويات

**الأساسيات**
**مكونات ++C وادواتها**
# Basic Elements of C++

## رموز لغة ++C

*الرموز المستخدمة في لغة ++C

١- الحروف الإنجليزية الكبيرة A.B.C
٢- الحروف الإنجليزية الصغيرة a.b.c
٣- الأرقام العربية الأصل 1.2.3
٤- رموز خاصة مثل:

| | | | | | |
|---|---|---|---|---|---|
| [] | " | ! | < | - | + |
| * | ، | \|\| | > | ( ) | _ |
| >> | <> | <= | >= | \ | / |
| != | & | % | $ | # | << |

**الجدول ١-١**

وتعد هذه الرموز بأنواعها المادة الخام التي تتكون منها مفردات لغة ++C ، وإذا درست لغة أخرى قبل لغة ++C ، فانك تلاحظ أن لغة ++C ، تستعمل رموزا إضافية في لوحة مفاتيح الحاسب لا توجد في بعض اللغات.

*كلمات لغة ++C
الكلمات نوعين:-
١- أسماء تعريفية (Identifiers)
وهي الأسماء التي نسميها نحن " المبرمجون" تعرف الحاسوب بما تريد.

وتطلق الأسماء التعريفية على:-
A- المتغيرات.
B- الاختزانات (الدوال).
C- المؤشرات.

*قواعد تسمية الأسماء التعريفية في لغة C++ :-

١- أن يكون الاسم مكتوبا من سلسلة متصلة من الحروف أو الأرقام بشرط أن يبدا بحرف أو بخط تحتي "_"
٢- أن لا يحتوى الاسم على رموز خاصة عدا الخط التحتي "_"
٢- أن لا يكون الاسم إحدى الكلمات المحجوزة.

بعض الأمثلة الصحيحة على الأسماء التعريفية:

a. B6
b. X_ray
c. Matrix
d. Ok_
e. A
f. Soft_fine
g. Door12
h. _new

وهذه أسماء تعريفية غير مقبول (invalid) للأسباب المبينة إزاء كل منها:

7-up Ù لانه بدا برقم وليس بحرف.
b6.1 Ù لاستعماله الرمز الخاص (.)
salim! Ù لاستعماله الرمز الخاص (!)
Ђ2 Ù لا يجوز استعمال حروف غير إنجليزية.
No#1 Ù لاستعماله الرمز الخاص (#)

ومن الجدير بالذكر ، أن لغة C++ تفرق بين الحروف الأبجدية الصغيرة والكبيرة ، فمثلا الأسماء : System , system, SYSTEM تعامل كأسماء مختلفة عن بعضها البعض بسبب اختلاف معاملة المترجم للحروف الصغيرة والكبيرة.

٢- الكلمات المحجوزة

وهي كلمات قياسية معروفة مسبقا لمترجم ++C ، وتكتب عادة بحروف صغيرة ، ولها معان خاصة بها تؤديها في برنامج ++C ، وهذه الكلمات المحجوزة حسب الترتيب الأبجدي هي:

| | | | | | |
|---|---|---|---|---|---|
| near | Static | asm | Double | long | Sizeof |
| do | int | While | new | auto | else |
| For | This | Void | Delete | Goto | if |
| const | Entry | char | Class | Public | Case |
| Continue | Extern | struct | inline | float | Private |
| Virtual | Volatile | Frinde | enum | near | Static |
| cdecl | Default | inline | Overload | Unsigned | Typedef |
| Signed | Pascal | Operator | Switch | Template | Union |
| Register | Protected | far | Catch | char | Const |
| | | | | break | Return |

**الجدول٢-١**

وينبغى التنبيه إلى أن هذه الكلمات المحجوزة ، لا يجوز إعادة تعريفها أو استعمالها لغير ما خصصت له.

وكما تلاحظ من قائمة الكلمات المحجوزة ، أن لغة ++C تعد لغة صغيرة إذ تتكون من عدد قليل من الكلمات المحجوزة تقريبا ٥٢ كلمة محجوزة فقط.

- تمثيل الثوابت العددية Numeric Constants

يمكن تمثيل الثوابت العددية ، في لغة ++C بثلاث صور هي:-

a. الثابت العددي الصحيح integer
  - هو عدد مكون من الأرقام من 0 Ù 9
  - لا يحتوى على فاصلة عشرية.
  - يمكن أن يحوى الإشارة "+" أو "-"

أمثلة صحيحة على الثابت العددي الصحيح:-

0
15
1000
321
-61

والأعداد التالية غير صحيحة للأسباب المبينة إزاء كل منها:

3.31 : لانه يحتوى على فاصلة عشرية.
1,000 : لانه يحتوى على فارزة.
J72 : لانه يحتوى على حرف أبجدي.
2 4 : لوجود فراغ بين العديدين.
1992 1992 1999 : لوجود فراغ وأيضا لان العدد كبير.

كما يمكن تصنيف الأعداد الصحيحة في لغة ++C ، حسب طولها ، والسعة التخزينية لها في الذاكرة مثلا:-
الثوابت الصحيحة 19897 , 40000 تسمى ثوابت صحيحة طويلة long int.
الثوابت 16- , 80 , 45 تسمى ثوابت صحيحة قصيرة short int.
الثوابت 20000 , 967 تسمى ثوابت صحيحة بدون إشارة unsigned int.

والفرق بين الثوابت الطويلة والقصيرة هو في عدد الوحدات التخزينية المطلوبة لكل نوع في الذاكرة ، فالطويلة تأخذ حيزا اكبر ، والقصيرة توفر عدد الوحدات التخزينية المستعملة ، أما الثوابت الصحيحة بدون إشارة unsigned int ، فان استعمالها يوفر وحدة تخزينية واحدة تستعمل للاشاره عندما تذكر كلمة unsigned ، قبل int ،

وذلـك بإزاحـة القيمة إلـى قيمة موجبة بـدون إشـارة ، ولكـل نـوع مـن الأنـواع السـابقة تطبيقاته المناسبة.

b- الثابت العددي الحقيقي Floating-point Constants

- هو عدد مكون من الأرقام 0 Ù 9
- يجب أن يحتوى على فاصلة عشرية
- يمكن أن يحوى الاشاره "+" أو "-"
- لا يجوز أن يحتوى على فارزة "،"

أمثلة على ثوابت عدد حقيقي تستعمل الفاصلة العشرية بشكل صحيح :-

421.5
10.6
0.0
0
01
-68.0

والأمثلة الاتيه غير صحيحة للأسباب المبنية إزاء كل منها:-

1000 : لانه لا يحتوى علي فاصلة عشرية.
4,000.21 : لانه يحتوى على فارزة.
2 83.4 : لان يحتوى على فراغ .

- تمثيل الثوابت الرمزية Non-numeric
  - سلسلة من رموز اللغة ( أحرف أرقام رموز خاصة) محصورة بين حواصر علوية مزدوجة (علامات تنصيص أو اقتباس)

ومن الأمثلة على الثابت الرمزي ما يأتي :-

"first"

"my name is"

"30+50=80"

"my,no=123.04"

"Islam"

وتلاحظ أننا سمينا أي نص موضوع بين حاصرتين مزدوجتين ثابتا رمزيا والصحيح أن تسميته ثابتا رمزيا هي من قبيل المجاز والاصطلاح لا الحقيقة ، واما كلمة رمزي : فلان النص مكون من عدد من الرموز ، وتسمية بعض الكتب بالثابت غير العدد .Non-numeric

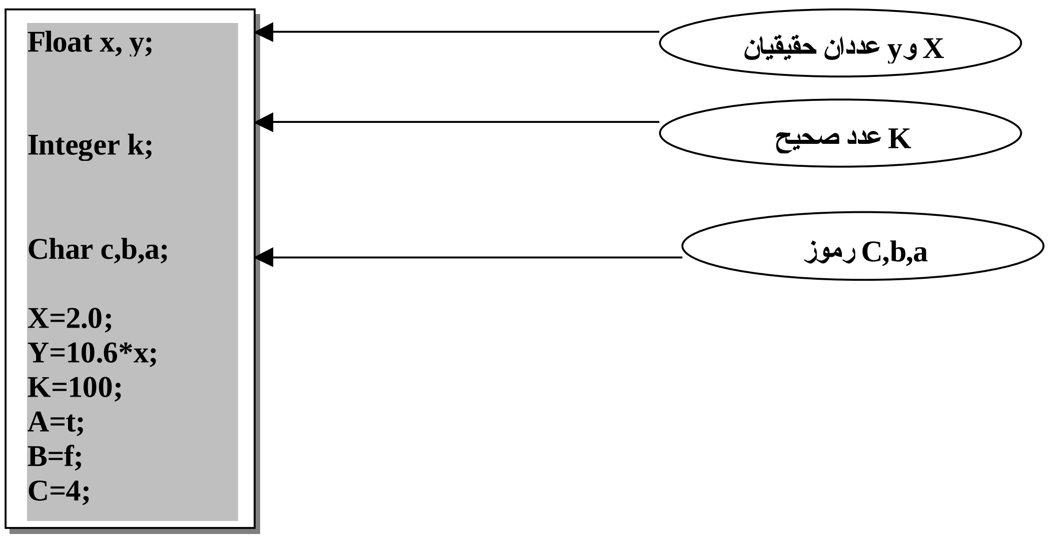
ملاحظة/

كل الثوابت الرمزية الواردة أعلاه ، وان استخدمت ارقاما حسابية داخلها ، ألا أنا لا تحمل أي قيمة حسابية ، وليس لها معنى حسابي ، وتستخدم مثل هذه الثوابت عادة كمعلومات توضيحية مع نتائج البرنامج.

# المتغيرات

هي أسماء (عناوين) لمواقع في ذاكرة الحاسوب ، يخزن بها رموز أو أعداد.

وبما أن أنواع المعلومات المراد تخزينها تكون عادة مختلفة مثل القيم الصحيحة أو الحقيقية أو الرمزية ... الخ فانك تحتاج أن تعلم المترجم في بداية البرنامج عن أنواع المتغيرات التي تريد استعمالها في البرنامج ، فمثلا في السطور التالية تبين أن المتغيرين x و y حقيقيان ، والمتغير k صحيح ، والمتغير c,b,a رمزية.




لاحظ السطرين الأول ، والثالث يحتويان على اكثر من متغير حيث يفصل بين كل متغيرين ، فاصلة وكذلك يمكن تحديد أنواع المتغيرات ، بذكر التفصيل الدقيق للنوع ، من حيث طول السعة التخزينية ، أي هل هو صحيح قصير أم طويل حقيقي قصير أو مضاعف ... الخ

- وتقسم المتغيرات لنوعين :-

١- متغيرات عددية

وهي مواقع في الذاكرة تخزن بها أعداد .

٢- متغيرات رمزية

وهي مواقع في الذاكرة تخزن بها رموز.

٣- متغيرات منطقية

وتخزن بها قيمة منطقية أما 0= FALSE أو 1=TRUE

# الأدوات المستعملة في لغة ++C

يوجد في لغة ++C ثلاثة أنواع من الأدوات وهي:
الأدوات الحسابية – الأدوات المنطقية والعلاقة – الأدوات الدقيقة وفيما يأتي تفصيل بالأنواع الثلاثة:-

## الأدوات الحسابية Arithmetic Operators

تسمح لغة ++C باستعمال الأدوات الحسابية من جمع وطرح وضري وقسمة ، كاللغات الأخرى ، إلا أن عملية الرف إلى أس ، ليس لها أدوات مباشرة مثل الأداة h في Basic والأداة ** في فورتران ، وانما تتم علمية الرفع إلى أس في لغة ++C بطريقة أخرى ..
كما تختلف القسمة في لغة ++C عنها في Basic إذا أن أي جزء كسري ينتج عن القسمة يهمل مهمها كان كبيرا ، كما في لغتي باسكال وكوبول فمثلا ناتج القسمة 8/3 هو 2 لا الكسر 0.666 يهمل ، ويكون ناتج القسمة باستعمال الأداة / صحيح العدد.
ويمكننا الآن أن نلخص الأدوات الحسابية المستعملة في لغة ++C فيما يأتي:-

| وظيفتها | الأداة |
|---|---|
| للطرح أو كأشاره سالبة | - |
| للجمع | + |
| للضرب | * |
| للقسمة | / |
| لباقي القسمة الصحيحة | % |
| للنقصان | - - |
| للزيادة | + + |

**الجدول٣-١**

ويختلف أداء بعض الأدوات الحسابية حسب نوع المعطيات الصحيحة ، أو الحقيقة ، أو الرمزية فعند معاملة المعطيات الحقيقية للأدوات الحسابية ، يمكن القول أن العلميات الأساسية من جمع وطرح وضرب ، تجري بالطريقة التي نعرفها ، إلا أن هناك محذورا يجب أن نذكر به ، وهو أن تتعدي قيمة النتيجة من أية عملية حسابية الحدود المرسومة لنوع المتغير الناتج ، لان لكل نوع من أنواع المتغيرات حدودا ، يعد تجاوزها خطأ ينتج عنه خطأ في النتائج ، وعند معاملة المعطيات الصحيحة بالأدوات الحسابية تعمل الأدوات بالطريقة التي نتوقعها ، وعند تعدي الحدود المسموح بها في القيم الصحيحة ، فان هذا يعني أن خطأ قد وقع overflow ، وفي هذه الحالة لن تتلقى من المترجم أية رسالة خطأ ، فمثلا إذا كان لدينا البرنامج التالي:

```
Main( )
{
int n = 33000;
n = n * 3;
}
```

عند طباعة النتيجة n النهائية نتوقع أن يكون الجواب 99000 ، ألا أن الجواب في هذه الحالة لن يتعدى 30464 ، وهو الحد الاعلى المسموح به للقيمة الصحيحة ، وهناك أمر أخرى يتعلق بالقسمة فعندما نقسم 8 على 3 قسمة صحيحة 8/3 فان الناتج يكون صحيحا وهو 2 فقط ، وإذا ما رغبت أن تحافظ على الجزء الكسري الذي أهمل واسقط ، يمكنك أن تحول القسمة إلى قسمة حقيقة 8.0/3.0 حينئذ فان الناتج سيكون 2.667 لهذا السبب أدخلت لغة ++C أداة باقي ا لقسمة % ويسمى Modulus Operator ويستعمل على النحو التالي:

```
7 % 3
```

تعطي الجواب 1 وهو باقي القسمة الصحيحة 7/3 ، ومن الجدير بالذكر أن كلا من باسكال وكوبول تستعملان مثل هذه العملية ، ففي باسكال تكتب هذه العملية على النحو 3 mod 7 ، وكلمة MOD هي اختصار Modulus ، أما في لغة ++C فتستعمل الأداة % لتقوم بهذا العمل.

## الأدوات الأحادية والثنائية Unary and Binary Operators

تعد جميع أدوات الجمع والطرح والضرب والقسمة وباقي القسمة أدوات ثنائية binary أي أنها تأخذ (تتعامل مع ) قيمتين وتنتج قيمة واحدة ، فمثلا نتيجة 3*2 هي القيمة 6 وهناك الأداة الأحادية – عندما تتعامل مع قيمة واحد فمثلا (1992-) تمثل الإشارة ا لسالبه وهي هنا أداة أحادية Unary ، والعملية هنا ليست عملية طرح كما نعلم.

## الزيادة والنقصان Increment and Decrement

من مزايا لغة C++ أنها تستعمل الأداتين الحسابيتين + + و - - لزيادة القيم بمقدار 1 أو إنقاصها بمقدار 1 ، والمثال التالي يبين طريقة الاستعمال:

```
A++;
```

```
++a;
```

معناه إضافة قيمة 1 إلى a ويمكن كتابتها بصوره مكافئة على النحو التالي:-

```
A=a+1;
```

وبالطريقة نفسها يمكن إنقاص 1 من قيمة a على النحو:-

```
--a;
```

أو

```
a--;
```

وهو يكافئ الصورة

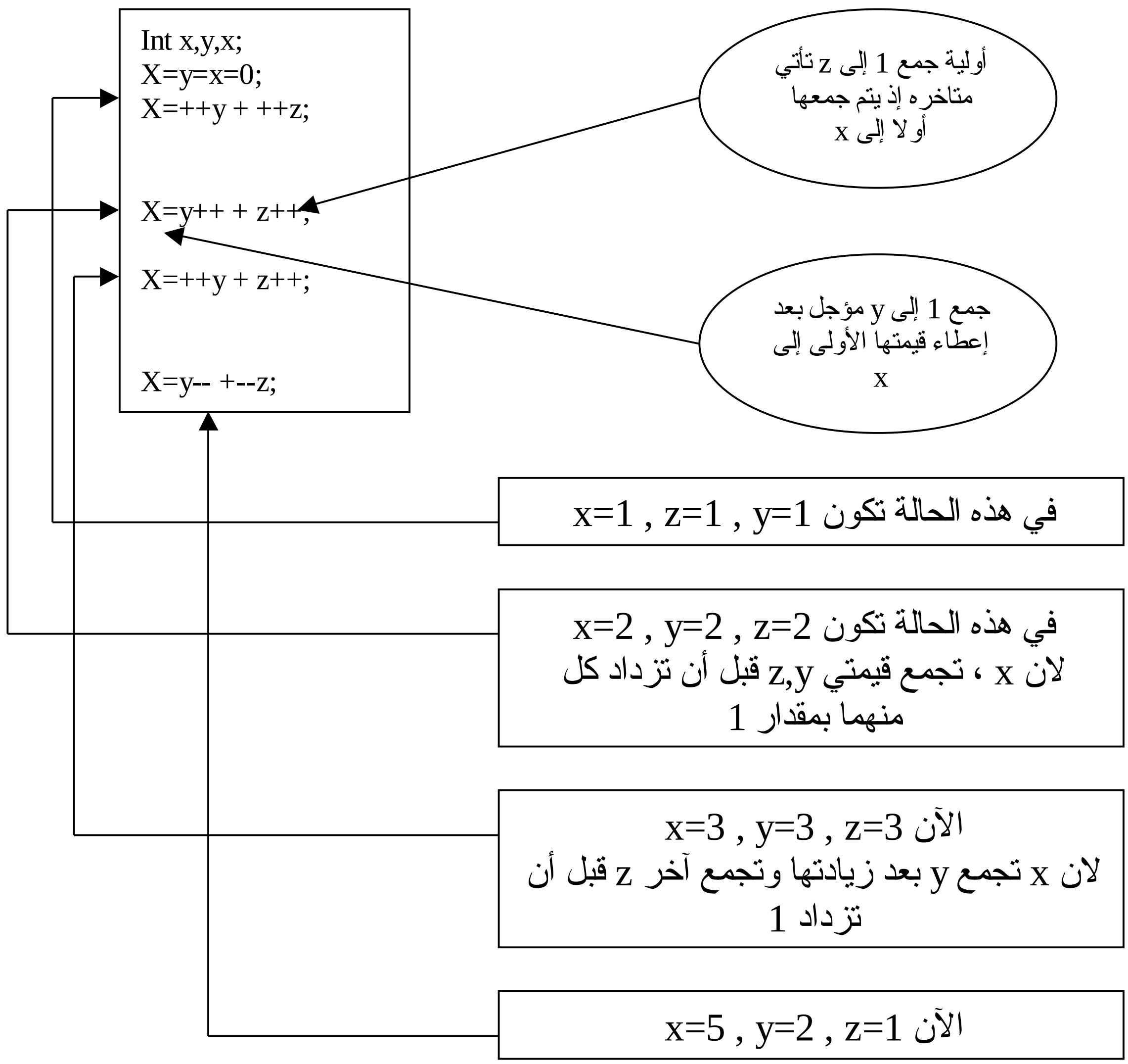
```
A=a-1;
```

لكن هناك فرقا في سرعة التنفيذ ، فالتعبير ;++a أسرع من التعبير ;a=a+1 وهذه هي الفائدة من جراء استخدام مثل هذه الأدوات .

ومما ينبغي التنبيه إليه هنا أن هناك فرقا بين ++a و a++ ، صحيح أن كلا من التعبيرين يجمع 1 إلى a ، لكن عند استعمال ++a في تعبير من التعابير ، فان a

تـزداد قـبل اسـتخراج قـيمة التعبـير ، بيـنما فـي حالـة a++ تسـتخرج قـيمة التعبـير باستعمال قيمة a الحالية قبل زيادتها بمقدار 1 ، وبعد ذلك تتم زيـادة a بمقدار 1 آي أن العملية الأولـى جمع تقديم ، والثانـية جمـع تأخير ، وينطبق هذا الكلام أيضا على – a و --a .

مثال:




وبإمكانك كتابة الجملتين:

Int x,y,z
X=y=z=0

في جملة واحد على النحو:

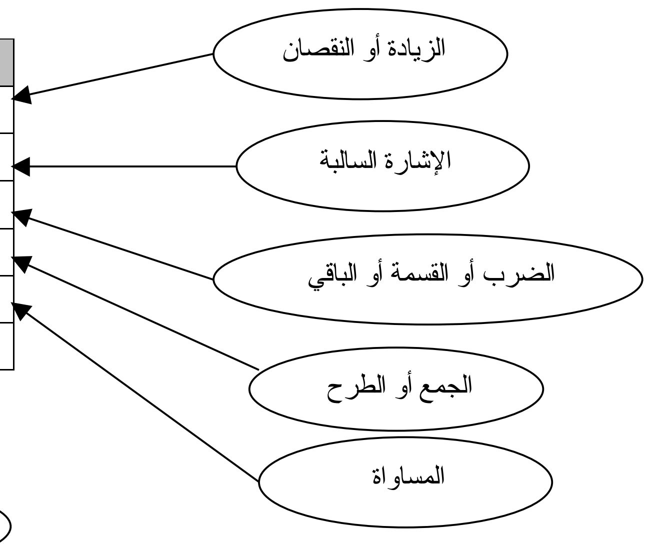
Int x=y=z=0

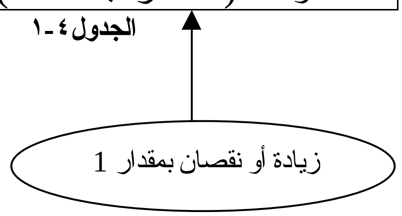
# أولية العمليات للأدوات الحسابية Precedence of Arithmetic Operations

يمكـن القـول أن أولـية تنفـيذ العملـيات كمـا يجـريها مـترجم ++C بالنسـبة لـلأدوات الحسابية هي على النحو التالي:

| رقم الأولوية | الأداة |
|---|---|
| 1 | ++ أو -- |
| 2 | - |
| 3 | * أو / أو % |
| 4 | + أو - |
| 5 | = |
| 6 | ++ أو -- (المتأخرة بعد العدد) |

الجدول ٤-١




ملحوظة:

إذا تساوت أوليتان مثل الجمع والطرح في تعبير ، فتقدم العملية الأقرب إلى يسار التعبير ، وعند استعمال الأقواس لأي تعبير فان الأقواس تأخذ الأولوية الأولى في التنفيذ قبل ( الزيادة أو النقصان ) ، كما في لغات البرمجة الأخرى ، والأمثلة الآتية تبين مفهوم الأولوية ( الأسبقية ):-

```
X + y / z * a
```

يأخذ تسلسل أولويات عملياته الشكل والخطوات التالية:-

١- العملية الأولى: القسمة y / z

٢- العملية الثانية: a * (y/z)

٣- العملية الثالثة: جمع الناتج في الخطوة 2 إلى x فتكون النتيجة:

X + y / z * z

لاحظ أننا بدأنا بإجراء العلميات الحسابية من اليسار إلى اليمين ، وتعطى ألا ولويه لأية عملية حسب قاعدة الأولوية ، فجاءت القسمة ، في المثال قبل الجمع ، كما جاء الضرب بعد القسمة وتلا ذلك الجمع كأخر عملية.

# الأدوات العلاقية والمنطقية Relational and Logical Operations

يرجع اسم الأدوات العلاقية إلى العمليات المختصة بالقيم التي بينها علاقات وهو أجراء عمليات مقارنة منطقية بين كميات حسابية أو رمزية ، وتكون نتيجته منطقية وهي أما نعم (true) أو (false) ، ويكثر استخدام التعابير المنطقية في الجمل الشرطية ، والأمثلة الآتية تبين لك ما هو التعبير المنطقي:

التعبير المنطقي: x= =y جواب أما نعم أو لا .
والتعبير المنطقي: matrix > 100.0 جواب أما نعم أو لا .

وفي لغة ++C تعامل النتيجة لا (false) على أنها صفر (0) وتأخذ النتيجة نعم (true) أية قيمة غير الصفر والمشهور أنها (1) .
ويبين لنا الجدول التالي الأدوات العلاقية والمنطقية:

الأدوات العلاقية

| الأداة | معناها |
|---|---|
| > | اكبر من |
| < | اصغر من |
| >= | اكبر من أو يساوي |
| <= | اصغر من أو يساوي |
| = = | يساوي |
| != | لا يساوي |

**الجدول ٥-١**

الأدوات المنطقية

| الأداة | معناها |
|---|---|
| && | And (حرف العطف واو) |
| \|\| | Or ( حرف العطف أو ) |
| ! | Not (للنفي) أداة أحادية unary |

**الجدول ٦-١**

إليك الآن هذه الأمثلة : افرض أن ;int a=b=3
فان التعبير a < 3 نتيجته false أي 0
التعبير a<=3 نتيجته true أي 1
التعبير a>b نتيجته false أي 0
التعبير a!=b نتيجته false أي 0
التعبير a= = b نتيجته true أي 1

<u>جدول الصدق</u> سوف نسوق هذا الجدول كالتالي:-

<table>
<tr><th colspan="2">جدول النفي<br>!x (not x)</th><th colspan="3">جدول التخيير<br>X||y (x or y)</th><th colspan="3">جدول الجمع<br>X && y (x and y)</th></tr>
<tr><td>x</td><td>!y</td><td>x</td><td>Y</td><td>X||y</td><td>X</td><td>Y</td><td>X&&y</td></tr>
<tr><td>F</td><td>T</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td></tr>
<tr><td>T</td><td>F</td><td>F</td><td>T</td><td>T</td><td>F</td><td>T</td><td>F</td></tr>
<tr><td colspan="2" rowspan="2"></td><td>T</td><td>F</td><td>T</td><td>T</td><td>F</td><td>F</td></tr>
<tr><td>T</td><td>T</td><td>T</td><td>T</td><td>T</td><td>T</td></tr>
</table>

**الجدول ٧-١**

!! المساعدة على فهم جداول الجمع والتخيير والنفي أعلاه:-

جدول الجمع:

تخيل أن F تمثل السم ، وان T تمثل العسل ، وبناء على ذلك فان F&&T تعني سما مع سم والنتيجة سم أي F ، كذلك F&&T تعني خلط السم مع العسل والنتيجة سم أي F ، وكذلك T&&F ينتج عنها F أما T&&T فهي عسل على عسل أي أن النتيجة T .

جدول التخيير:

فلو خيرت بين السم F والسم F F||F فالنتيجة معروفه F أما بين السم والعسل F||T فالنتيجة سوف تكون بالطبع للنجاة عسل T ، ونتيجة T||T هي عسل T...

# الأدوات الدقيقة Bowties Operators

تتميز لغة ++C عن سائر اللغات الراقية مثل فيجوال بيسك وباسكال أنها تستخدم أدوات دقيقة على مستوى وحدة التخزين الأولية [Bit] والمختصرة من Binary Digit*

سميت هذه الأدوات بالدقيقة أو أدوات (البت) لأنها تتعامل مع [bit] ( وحدة التخزين الأولية) مباشرة ، فحصا ، وضبطا ، وإزاحة ، وتستعمل هذه الأدوات مع المعطيات الصحيحة int والرمزية char فقط ، ولا تستعمل مع غيرها من أنواع المعطيات..

والجدول التالي يبين الأدوات الدقيقة ووظيفة كل منها:

| الأداة | عملها |
|---|---|
| ~ | آداه أحادية (not) |
| & | حرف الواو (و) (and) |
| \| | حرف العطف (و) (or) |
| >> | إزاحة إلى اليسار |
| << | إزاحة إلى اليمين |
| ^ | (أو) الاستثنائية (xor) |

الجدول ٨-١

وكغيرها من الأدوات ، فان الأدوات الدقيقة تتبع قواعد الأولية وحسب الترتيب التالي:

| الأداة | أولويتها |
|---|---|
| ~ | الأولى |
| << أو >> | الثانية |
| & | الثالثة |
| ^ | الرابعة |
| \| | الخامسة |

الجدول ٩-١

## أداة النفي (~)

تعمل هذه الأداة على إبدال الصفر (0) بواحد(1) أو العكس ، ومعنى هذا أنها تضع 0 مكان 1 وكذلك 1 مكان 0 ، فمثلا لو كان لدينا قيمة x ممثلة في النظام العددي الثنائي التالي (من 8 بت ):-

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| قيمة x | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| قيمةx~ | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |

ومعنى ~ النفي (not) ومعنى النفي هنا التضاد بين 0 و 1 في النظام العددي الثنائي ، فعندما تنفى 0 تثبت بدلا منه 1 والعكس صحيح ، وهذا يوضحه لك المثال السابق إذ تم (نفي) قيمة x بالبت ليصبح x~ في جميع مكونات من البت.

## أداة الجمع &

المثالي التالي يوضح كيفية جمع القيم عند تمثيلها بالنظام العددي الثنائي:

العملية ;x&y

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | قيمة x بالنظام الثنائي |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | قيمة y بالنظام الثنائي |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | الناتج ;X&Y |

حيث يجمع 0+0 ويعطي 0 ، ويجمع 1+0 ليعطي 0 ويجمع 1+1 ويعطي 1

F Ù F&F    F Ù F&T    T Ù T&T (انظر جداول الصدق السابقة)

## أداة الاختيار|

إذا أردنا استعمال أداة الاختيار مع المثال السابق لقيمتي X و y على النحو ;x|y

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | x |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | y |
| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | الناتج ;x\|y |

حيث الاختيار بين 0 و 1 هو 1 ، والاختيار بين 1 و 1 هو 1 ، وبين 0 و 0 هو 0 .

T Ù F|T   T Ù T|T   T Ù T|F ( انظر جداول الصدق السابقة )

# <u>أداة الاختيار الاستثنائي ^</u>

خذ المثال السابق نفسة لترى كيف تؤثر هذه الأداة

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | x |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | y |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | الناتج x^y; |

لاحظ أن الاختيار بين 0 و0 نتيجته 0 ، والاختيار بين 1 و0 نتيجته 1 ، وبين 1 و1 نتيجته 0 ، أن أداة الاختيار الاستثنائي ^ (XOR) تختلف عن أداة الاختيار (OR) | في أن الأولى حالة خاصة من الثانية إذ تكون نتيجتها true (1) عندما يكون أحد طرفي الاختيار x أو y نتيجته true ، وما عدا ذلك فنتيجتها دائما false (0) ، كما لاحظت في هذا المثال ، أما في حالة الاختيار العام (OR) فان نتيجة 1|1 هي 1.

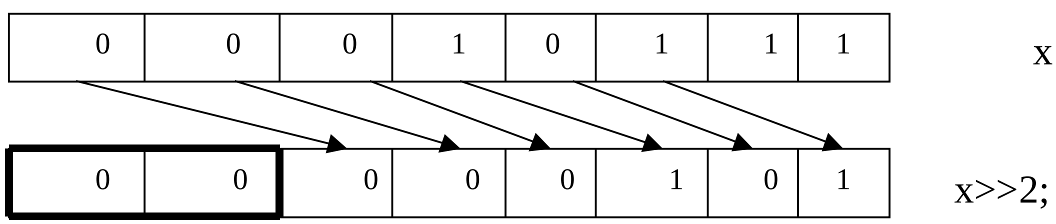
## أدوات الإزاحة >> و <<

قد تبدو أدوات الإزاحة غريبة على المبرمج الذي يستخدم لغات أخرى مثل Basic و Pascal .. الخ . حيث ينتج عن استعمال إحداهما إزاحة قيمة المتغير الصحيح بالنظام الثنائي (بالبت) يمينا أو يسارا عددا من الخانات حسب الطلب ، وتملا الخانات المفرغة من الجهة الموجبة أصفارا ، ومن الجهة السالبة تملا آحادا.

والأمثلة التالية توضح طريقة الاستعمال.

مثال:

الجملة ;x>>2 عند تنفيذها على قيمة x ( وهي 23 بالنظام العشري) بالنظام الثنائي فان العملية تتم على النحو التالي:




النتيجة من الإزاحة بمقدار خانتين (٢ بت ) لليمين تصبح قيمتها :

5 بالنظام العشري.

وهذا يعني أن ;23>>2 تعطي النتيجة 5 .

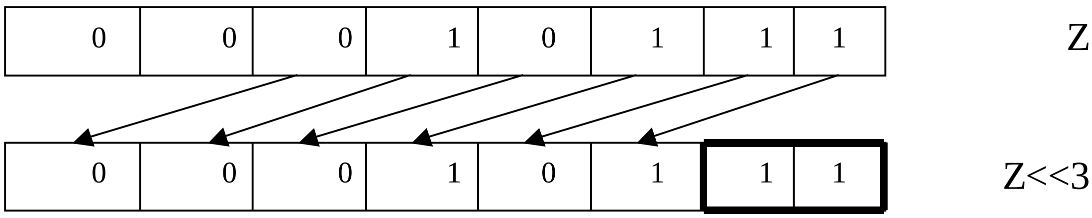
حيث 23 القيمة المزاحة.

2 عدد خانات الإزاحة المطلوبة لليمين.

لاحظ أن الخانتين المفرغتين بسبب الإزاحة لليمين قد ملئتا بمصفرين.

مثال على إزاحة قيمة سالبة:

Z = - 50<<2;




إزاحة (٢بت) لليسار.

الجديد في هذا المثال أن الإزاحة لقيمة سالبة ينتج عن كل خانة مفرغة القيمة 1 وليس 0 كما في المثال السابق.

# Other Operations أدوات أخرى لم تذكر
## the conditional operator الأداة الشرطية

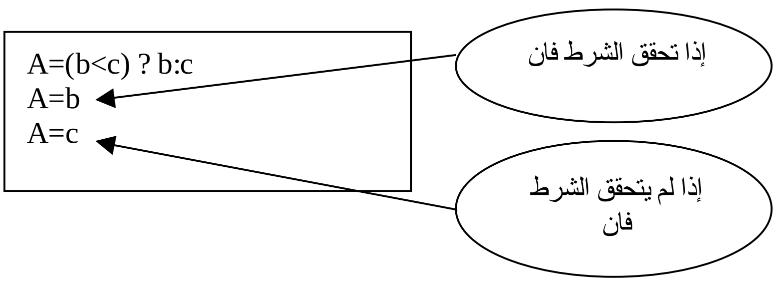
وهذه الأداة تتميز بها لغة ++c عن غيرها ، إذ تحل محل جملة شرطية مشهورة في بيسك وفورتران وباسكال وهي if-then-else ، وتعتبر هذه الأداة أداة ثلاثية لأنها تتعامل مع ثلاث كميات حسب صيغتها العامة التالية:

Expression1 ? Expression2: Expression3

فلو كان لدينا الجملة الشرطية التالية على سبيل المثال:

```
If (b<c)
A=b;
Else
A=c
```

معناها: انه إذا تحقق الشرط (b<c) فينفذ السطر a=b ، وإلا فان a=c وباستعمال الأداة الشرطية ؟ يمكننا أن نكتب بدلا من الجملة الشرطية كلها الجملة المختصرة التالية:



## أداة العنونة (&) و (*) the & and * Pointer Operator

المؤشر Pointer هو عنوان المتغير المؤشر في الذاكرة ، وللمتغير المؤشر فوائد جمة في عملية البرمجة نذكرها عند التعرض لها في الفصول القادمة بإذن الله ، ويكثر استعمال هاتين الدالتين مع المتغيرات المؤشرة المحجوزة لها في أماكن خاصة في الذاكرة .

وتعد الأداة & هنا أداة أحادية تتعامل مع كمية واحدة فقط ، حيث تقوم بإعطاء عنوان الطرف الأيمن للمعادلة ، للطرف الأيسر منها ، فمثلا العملية:

```
X=&y;
```

تعطي عنوان y في الذاكرة ، ووضعه في مخزن x ، وهذه الجملة تختلف طبعا عن الجملة الحسابية x=y التي معناها وضع قيمة y في مخزن x أما عند استعمال & قبل y فمعناها إعطاء عنوان مخزن y في الذاكرة فقط للمتغير x ، وليس قيمة y فلو كانت y=10 ، ورقم (عنوان) موضعها في الذاكرة هو 120 ، فان x تأخذ قيمة 120 عند استعمالنا & مع y وبالنسبة للأداة الثانية * فهي أداة أحادية أيضا ، ومكملة للأداة & ، ولذلك لو كان لدينا الجملة التالية التي تستعمل الأداة * .

```
S=*x;
```

فانه يفهم منها أن x تحوى عنوان (موقع) المتغير y في الذاكرة ، وان هذه الجملة تضع في مخزن s قيمة المتغير ، صاحب المخزن الأصل ، آي قيمة y وهي 10 كما في المثال السابق ، وبالتالي فان قيمة 10 تخزن في مخزن s في الموقع (العنوان) 120 لذلك نرى أن جملة x=*&y تكافئها الجملة x=y وهذا يعني أن الأداتين تعملان وكأن الواحدة معكوسة للأخرى

ومما يجب الانتباه إليه ، أن الأداة * تستخدم أيضا لعمليات الضرب الحسابي كما تستعمل الأداة & كأداة AND دقيقة ، ولذا لا يلتبس عليك الأمر بين الاستعمالين المختلفين.

## أداة تعيين الطول sizeof

تعد هذه الأداة أداة أحادية (unary) ، وتستعمل لتعيين طول المتغيرات ( بالبايت) ، وتختلف أطوال المتغيرات حسب أنواعها ، ولذا طلب تعيين طول متغير باستعمال sizeof ، ينبغي ذكر نوع هذا المتغير بين قوسي sizeof، فمثلا:

```
Int n;
N=sizeof (n);
```

حيث ستكون نتيجة n هنا تساوي 2 بايت ، هي طول المتغير n الصحيح (int) لان طول الصحيح عادة هو 2 بايت ، وطول الحقيقي 4 بايت ، كما في المثال التالي:

```
Float x;
Z=sizeof (x);
```

حيث ستكون نتيجة z هي 4 بايت ، وهي طول x الحقيقي.

# الفاصلة (,) كأداة The Comma Operator

وهي أداة استثنائية (binary) وتحتل الأولوية الأخيرة في سلم الأداة المختلفة وتأخذ الصيغة العامة التالية:

Experssion1, Experssion2

فعندما تفصل فاصلة بين تعبيرين فان تسلسل العمليات يأخذ الترتيب التالي:

١- تستخرج قيمة التعبير الأول ( علي يسار الفاصلة ) ثم تعطى للتعبير الثاني.
٢- تستخرج قيمة التعبير الثاني ( علي يمين الفاصلة ) كقيمة نهائية للتعبير كله كما في المثال التالي:

```
A=(b=2,b+1);
```

حيث يعطى المتغير b قيمة 2 في التعبير الأول ( يسار الفاصلة) ، ثم وضع هذه القيمة في b الأخرى في التعبير الثاني ( يمين الفاصلة) ، فتصبح قيمة التعبير على اليمين (b+1) تساوي 3 وتكون هذه القيمة نتيجة التعبيرين النهائية .

مثال آخر:

```
B=8;
A=(b=b-4,12/b);
```

في هذا المثال يتم إعطاء b القيمة 8 أولا ، ثم عند تنفيذ السطر الثاني ، يعطي b في التعبير الأول داخل القوسين القيمة (b-4) آي (8-4) ، وتساوي 4 ، وهذه تعطى للتعبير الأيمن ، حيث تتم القسمة (12/b) آي (12/4) فتصبح نتيجة التعبير كله 3 ، التي تعطي بالتالي للمتغير a .

# جمل التعريف

جمل التعريف هي جمل تقوم بتعريف القيم.

مثال:

```
Int a;
```

يقابل هذه الجملة في فيجوال بيسك

```
Dim a as integer
```

وتقوم بحجز مكان في الذاكرة المشار إليه ، بالاسم a لتخزين قيمة عددية صحيحة.

أنواع البيانات الممكن تخزينها في الذاكرة المستخدمة لـ C++

١. char لتخزين رمز واحد فقط.
٢. int لتخزين عدد صحيح.
٣. float لتخزين عدد حقيقي.
٤. double لتخزين عدد حقيقي كبير.
٥. void لتخزين بيانات خالية.

أن معرفة أنواع البيانات ، وكيفية استعمالها ، تعد ضرورية لفهم لغة c++ فلاستعمال المتغيرات ، مثلا ، نحتاج أن نعلن في بداية كل برنامج ، أو بداية الدوال عن أنواع هذه المتغيرات ، ويتم التعامل معها ، خلال البرنامج ، في ضوء أنواع معطياتها فمثلا الإعلان عن التالية:

```
Int a,b,x;
```

تخبر مترجم c++ أن يتعامل مع هذه المتغيرات ، على أنها متغيرات صحيحة وكذلك جملة الإعلان التالية:

```
Float m,y;
```

تخبر مترجم c++ (C++ compiler) أن هذه المتغيرات من النوع الحقيقي.

## الثوابت الرمزية ذات الشرطة المعكوسة

حيث أننا لا نستطيع استعمال بعض الرموز الموجودة في لوحة مفاتيح الحاسب كثوابت رمزية ، فقد استحدث لغة c++ شفرات رمزية خاصة تستعمل شرطة معكوسة لها ، وهذه الشفرات مدونة في الجدول التالي:

| الشفرة | معناها | القيمة الصحيحة لها |
|---|---|---|
| "\b" | رجوع بمقدار خانة واحدة | 8 |
| "\n" | سطر جديد | 13 |
| "\t" | ترتيب أفقي | 9 |
| "\0" | للقيمة الخالية | 0 |
| "\r" | علامة رجوع | 13 |
| "\v" | ترتيب عمودي | 11 |
| "\\" | الشرطة المعكوسة \ | 92 |
| "/f" | تقديم صفحة | 12 |

**الجدول ١٠-١**

ولبيان أهمية هذه الشفرات ، خذ المثال التالي:

"first line\n second line"

لو طبع هذا النص (الثابت الرمزي) فانه سيظهر في سطرين متتالين على النحو التالي:

First line
Second line

ومن الجدير بالذكر ، أن أهم تطبيقات المعطيات الرمزية واستعمالاتها ، هو معالجة النصوص ، وما يستحق التسجيل والاهتمام ، انه يمكن إجراء عمليات على المعطيات الرمزية.

# الملاحظات والتعليقات في ++Comments c

تستعمل سائر لغات البرمجة جملا للتعليقات والملاحظات ، وكذلك لغة ++C مثلا الجملة التالية:

```
10 rem this is Islam
```

هي جملة ملاحظ في لغة بيسك ، تقابلها جملة تعليق التالية في لغة ++C:

```
// this is Islam
```

التي توضح بعد شرطتين (خطين مائلين)
وتستعمل جمل التعليق ، في أي مكان من البرنامج لإبداء ملاحظة ما ، عند سطر ما في البرنامج ، ولا تعد جملة تنفيذية ، بمعنى أنها لو حذفت من البرنامج ، لا يؤثر فيه ذلك شيئا ، وعادتا ما يتجاهلها المبرمجين .

مثال: لاحظ جمل التعليق التالية:

```
this is comment//
/*an example on comment in c++ language */
/*main() /* start your program
{
int x,y,z; //this line declares type of x,y,z
}
```

ومن الجدير بالذكر هنا ، ما يأتي:

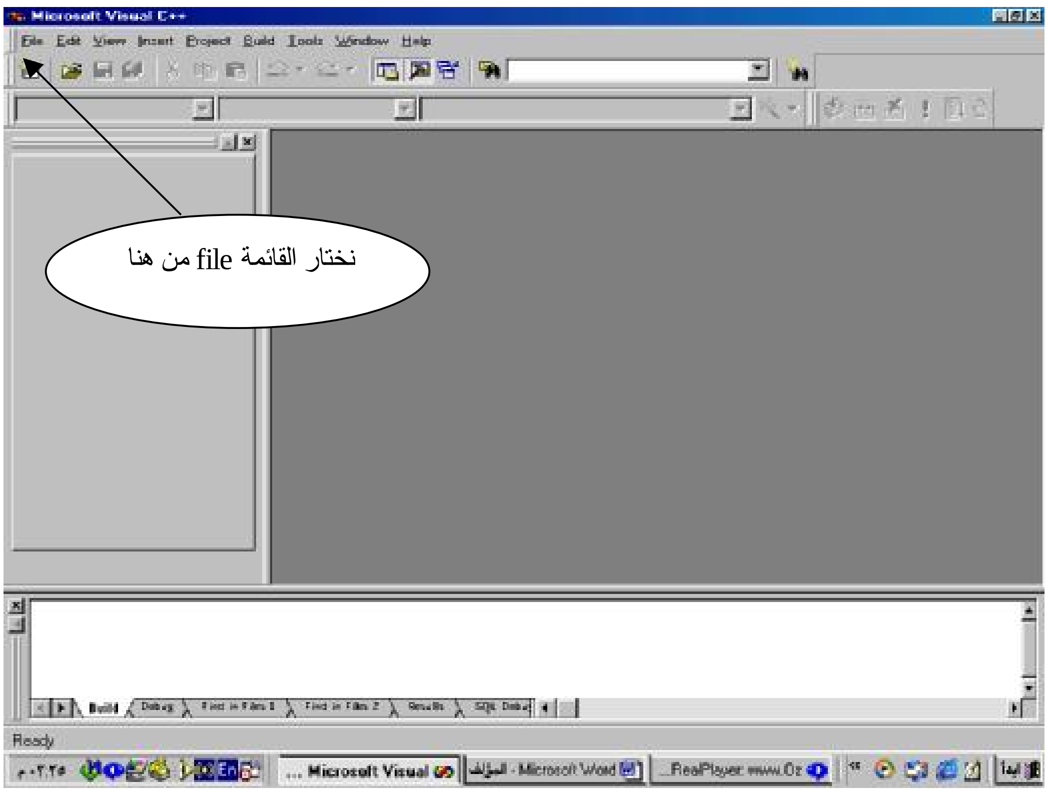
لا يترك أي فراغ بين الشرطة / والنجمة * من جهتي جملة التعليق.

١- يقوم مترجم ++c بإهمال النصوص المستعملة في جمل التعليق ، أي أنها لا تنفذ ، بل هي جمل توضيحية تظهر مع قائمة البرنامج أو سطورا فقط .

٢- يمكن وضع جملة الملاحظة والتعليق في أي مكان من البرنامج ، ما عدا وسط اسم تعريفي identifier ، أو كلمة محجوزة keyword .

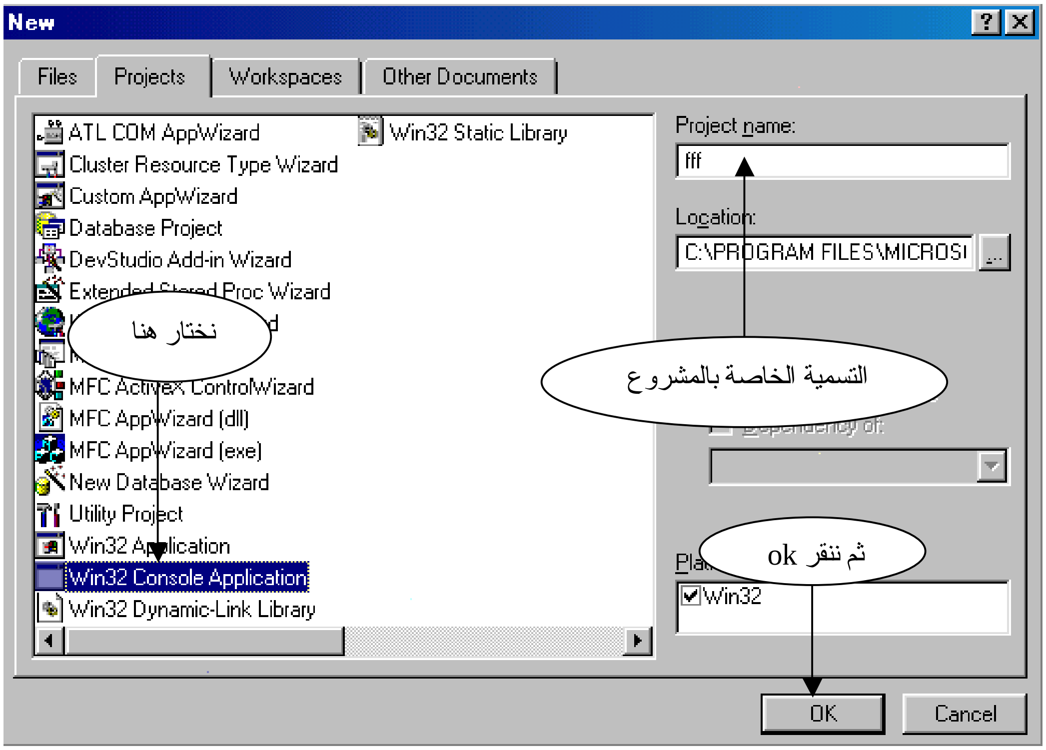
# تشغيل visual c++6.0
# Run visual c++6.0

لتشغيل برنامج فيجوال سي ++ نتبع التالي:
أبدا Ù البرامج Ù visualc++6.0
ثم بعد ذلك ستظهر لنا الشاشة التالية:

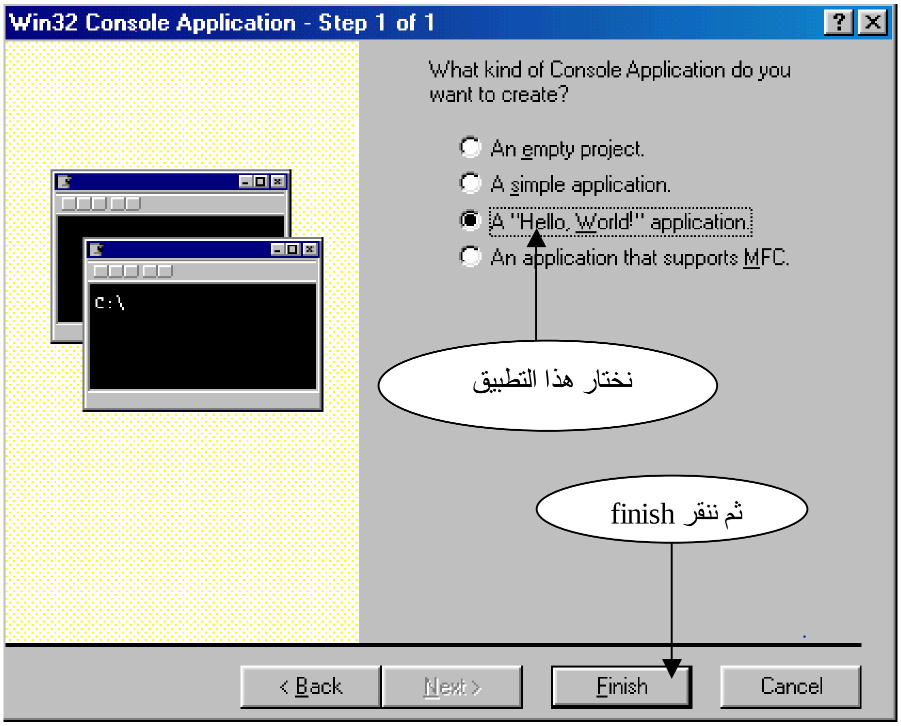



نختار من هذه الشاشة كما هو واضح القائمة File ثم بعد ذلك نختار من القائمة New لتظهر لنا الشاشة التالية..

نطبق ما يوجد بالصورة بالأسفل ثم نختار موافق..



نختار التطبيق الموجود بالأسفل ثم نختار إنهاء كما هو موضح بالأسفل..



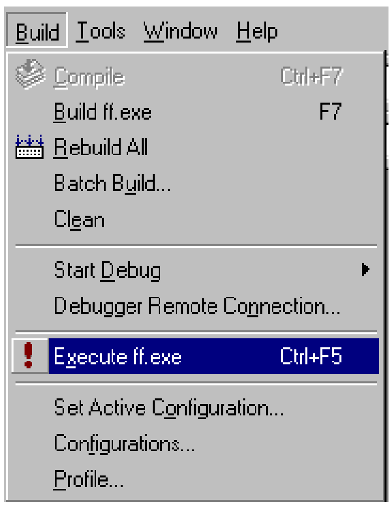
الآن ستظهر لنا شاشة الكود ونلاحظ بالأسفل الشاشة..

```
// fafa.cpp : Defines the entry point for the console application.
//

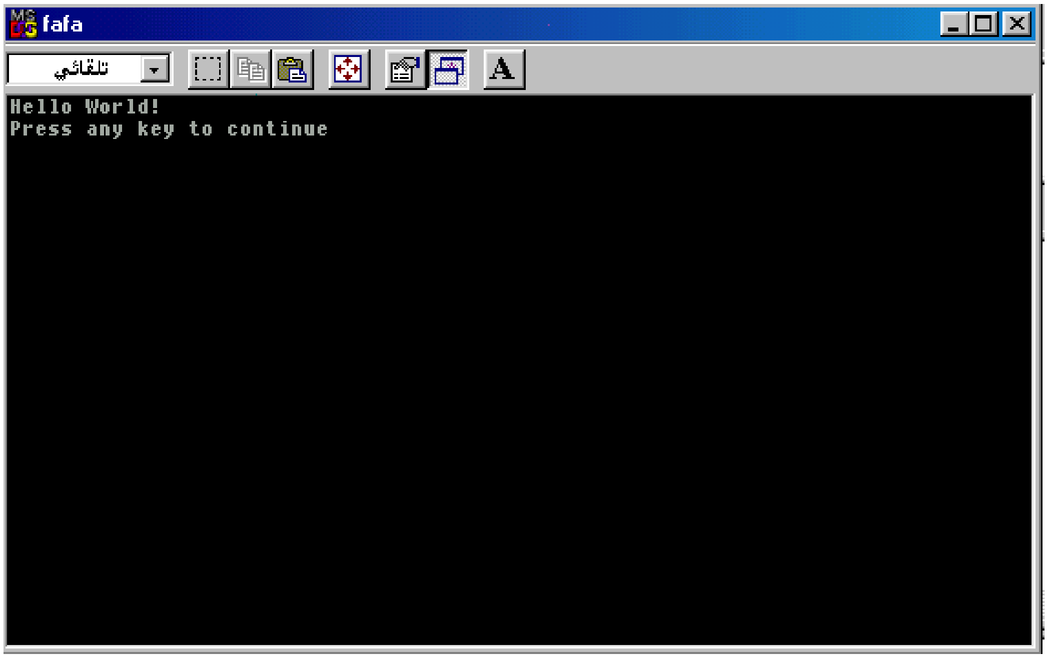
#include "stdafx.h"

int main(int argc, char* argv[])
{
	printf("Hello World!\n");
	return 0;
}
```

بعد ذلك نختار من القائمة Build ثم نختار exe. Execute ليطبق لنا المشروع ..
أو بالنقر من لوحة المفاتيح على الزر F5 .



طبعا بعد تنفيذ البرنامج ستظهر لنا النتائج كما في الشكل التالي:




طبعا أعزائي هذا البرنامج الصغير هو برنامج تلاحظون أن الكود تبعه خاص بلغة C الام وليس c++ لكن ما نعمل عليه هو مترجم يقبل اللغتين معا والمترجم هو Visual C++ ، ونلاحظ انه تم طباعة عبارة Hello World! وهي نتيجة تنفيذ الدالة printf() الموجودة في سطور البرنامج.

# أساليب الإدخال والإخراج
# Input/output Techniques

## مقدمة introduction

يتناول هذا الجزء أساليب إدخال القيم الحسابية والرمزية ، وأساليب إخراج وطباعة نـتائج الـبرامج والتعابـير الحسـابية والرمـزية ، وطـباعة المعطـيات المخـتلفة حسـب الحاجة.

لقد تعودنـا فـي لغـة بيسك ، أن نستعمل دوال مبنـية وجاهزة عند الطلب للقيام بالإدخال مـثل (input أو read) أو بالإخـراج مـثل (print) ، وفـي هـذا الصـدد ، فـان لغـة C++ ، تـتعامل مـع الإدخـال والإخـراج ، بطـريقة مخـتلفة، حيـث توفـر اللغـة ، عـددا كبـيرا من دوال الإخراج والإدخـال ، حيث يمكن للمبرمـج أن يستدعيها ، ويستفيد منها ، حسب نـوع المعطيات والمتغيرات ، كيفما يناسبه ، وسوف نورد أن شاء الله في هذا الفصل أهم هذه الدوال واشهرها لـ C++ .

# الإدخال والإخراج input\output

توفر لغة ++C ، مجموعة من الدوال والروتينيات المعرفة ضمن Iostream مثل cout للإخراج و cin للإدخال وسوف نعرف الملف iostream.h

الملف Iostream.h يعني:
io : مختصر لـ input/output آي الإدخال والإخراج.
Stream : مكتبة قياسية خاصة بالإخراج والإدخال الـخ..
H : header file آي الملف الدليل.

مثال ١:
إذا أردت طباعة العدد 100 في لغة بيسك فالجملة:

Print 100

تؤدي عملية الطباعة ، أما في لغة ++C فان الدالة التالية تعمل ذلك:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
cout<<100;
return0;
}
```

تؤدي إلى طباعة العدد 100 حيث cout اسم وحدة الإخراج القياسي والأداة<< تؤدي إلى إرسال العدد 100 إلى وحدة الإخراج ، أن هذا الأسلوب الجديد في الإخراج يختلف عما في لغة c .

# طباعة النصوص (الثوابت الرمزية)

مثال ٢:

تأمل قطعة البرنامج التالي:

```
#include  “ stdafx.h”
#include “ iostream.h”
main ()
{
cout<<“smoking is dangerous \n”;;
return0;
}
```

للانتقال لسطر جديد

بعبارة أخرى n\ إيعاز للانتقال إلى سطر جديد ، وقد يمكن استخدام الدالة endl بدلا من n\ وكما يلي:

```
#include  “ stdafx.h”
#include “ iostream.h”
main ()
{
cout<<“ smoking is dangerous”<<endl;
return0;
}
```

وعند تنفيذ البرنامج يطبع الناتج التالي:

smoking is dangerous

مثال ٣:
للاستفادة من إمكانيات الإيعاز \n في عمليات الطباعة :تأمل البرنامج التالي:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
cout<<"matrix";
cout<<"matrix \n";
cout<<"matrix \n\n";
cout<<"matrix \n\n\n";
cout<<"matrix";
return0;
}
```

عند تنفيذ البرنامج ترى الطباعة التالية على الشاشة:
السطر الأول Ù matrixmatrix
السطر الثاني Ù matrix
السطر الثالث Ù سطر فارغ
السطر الرابع Ù matrix
السطر الخامس Ù سطر فارغ
السطر السادس Ù سطر فارغ

نلاحظ في هذا البرنامج:
١- انه يتم الانتقال من السطر الأول بعد طباعة matrix إلى السطر الثاني لعدم وجود إيعاز الانتقال \n ، ولذا فان جملة الطابعة التالية ظهرت نتائجها في ا لسطر الأول نفسه ، متصلة بطاعة matrix الأولى ، وينتقل المؤشر الضوئي إلى سطر جديد لوجود إيعاز \n .
٢- يتم تنفيذ جملة الطباعة الثالثة في السطر الجديد (الثاني) ، ويتم الانتقال إلى السطر الرابع قفزا عن السطر الثالث ، وذلك لوجود الإيعاز \n\n حيث يقوم كل إيعاز \n بنقل المؤشر الضوئي سطرا واحدا ، وفي السطر الرابع تطبع جملة الطباعة الرابعة ، ويتم بعدها الانتقال إلى السطر السابع فورا حسب الإيعاز \n\n\n .

# طباعة القيم العددية

مثال٤:

يقوم البرنامج التالي بطاعة العدد 446 كقيمة صحيحة على شاشة الحاسوب:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
cout<<446;
return0;
}
```

عند الطباعة يظهر لنا التالي:

446

مثال٥:

برنامج ++C ، التالي يطبع القيمة الحقيقة 10.5:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
cout<<10.5;
return0;
}
```

عند الطباعة يظهر التالي:

10.5

مثال٦:

انظر ماذا يفعل برنامج C++ التالي:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int a=100;
cout<<a;
return0;
}
```

عند الطباعة يظهر لنا التالي:

100

مثال٧:

البرنامج التالي يقوم بطباعة قيمة متغير حقيقي:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
float x;
x=4.25
cout<<x;
return0;
}
```

عند الطباعة سيظهر لنا التالي:

4.25

مثال٨:

إذا تطلب الأمر طباعة المتغيرين a الصحيح ، و x الحقيقي الواردين في المثالين السابقين ، في برنامج واحد ، فالبرنامج سيكون على النحو التالي:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int=100;
float x=4.25
cout<<a<<x;
return0;
}
```

وستظهر نتائج هذا البرنامج كما طلبنا (الصحيح يسبق الحقيقي) ، هكذا:

100    4.25

# طباعة القيم العددية والرمزية (النصوص) في جملة واحدة

مثال ٩:

سوف نقوم في هذا المثال بطباعة قيم عددية ونصية مع البعض كالتالي:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int a=100;
cout<<"a="<<a;
return0;
}
```

عند الطباعة يكون الناتج كالتالي:

A=100

مثال ١٠:

ماذا إذا أردنا طباعة عدد صحيح وحقيقي مع نصوص بنفس الوقت:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int=100;
float x=4.25
cout<<"a="<<a<<"x="<<x;
return0;
}
```

وعند الطباعة سيظهر لنا التالي:

A=100  x=4.25

مثال ١١:
إذا أردنـا أن تظهـر نـتائج المـثال السـابق فـي سـطرين بـدلا مـن سـطر واحـد ، فجملـة الطباعة ستكون كالتالي:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int a=100;
float x=4.25
cout<<"a="<<a<<"\n"<<"x="<<x;
return0;
}
```

وتكون نتيجة الطباعة على الصورة التالية:

A=100
X=4.25

# الإدخال بلغة ++C  Streams

يتناول هذا المبدأ معالجة الإدخال حيث يعد استخدام streams افضل من دوال الإدخال للغة C .

وصيغة الجملة كالتالي:

```
Cin>>a;
```

ونشاهد أنها عكس عملية الإخراج حيث الإخراج >> أما الإدخال << .

وتستخدم هذه الجملة لإدخال قيم عبر لوحة المفاتيح للمتغيرات في الذاكرة ، ويتم تعيين قيمة المتغير في الذاكرة باستخدام لوحة المفاتيح .

ملاحظة/

لا يجوز أن نستخدم المتغير قبل تعريفة.

مثال صحيح:

```
Int x;
Cin>>x;
```

مثال خاطئ:

```
Cin>>x;
```

مثال١٢:

سوف نقوم بإدخال عدد صحيح في هذا التطبيق ثم نقوم بطباعته كالتالي:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int=a;
cin>>a;
cout<<a;
return0;
}
```

نلاحظ في هذا المثال أننا قمنا بتعريف المتغير a بأنه عدد صحيح بعد ذلك عند تنفيذ البرنامج سيطلب منا إدخال عدد سندخل العدد 10 مثلا عند ذلك سيكون الناتج كالتالي:

مثال١٣:

اكتب برنامجا لإدخال عمرك ثم طباعته ، وطباعه نصف وضعفه؟

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int a;
cin>>a;
cout<<a;
cout<<a/2;
cout<<a*2;
return0;
}
```

في المثال أعلاه قمنا أولا بتعريف المتغير كالتالي:

Int a;

ثم بعد ذلك طلب منا إدخال العمر:

عند الطلب سندخل مثلا 20

Cin>>a

وبعدها قمنا بطباعة العمر:

Cout<<a;

ثم قمنا حسب المطلوب بطباعة نصف العمر:

Cout<<a/2;

ثم قمنا حسب المطلوب الأخير بطباعة ضعف العمر:

Cout<<a*2;

لتكون النتيجة النهائية كالتالي:

20 10 40

مثال ١٤:

اكتب برنامجا لإدخال عدد ما وليكن العدد 7 ومن ثم طباعة جدول الضرب له؟

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int a;
cin>>a;
cout<<a*1;
cout<<a*2;
cout<<a*3;
cout<<a*4;
cout<<a*5;
cout<<a*6;
cout<<a*7;
cout<<a*8;
cout<<a*9;
cout<<a*10;
return0;
}
```

(عند طلب إدخال قيمة ندخل الرقم 7 حسب طلب السؤال..)

عند تنفيذ البرنامج ستكون النتيجة كالتالي:

7 14 21 28 35 42 49 56 63 70

مثال ١٥:

اكتب برنامج لإدخال ثلاث علامات لطالب 30 25 40 وطباعة معدل العلامات؟

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int a,b,c;
cin>>a>>b>>c;
cout<<(a+b+c)/3;
return0;
}
```

(ندخل حسب المطلوب 25 40 30)

نلاحظ أن في جمل الإخراج قمنا بكتابة قاعدة وهي جمع الثلاث أعداد مع بعضها ثم قسمتها على عددها وهي قاعدة معروفة لإظهار المعدل..

وسف يكون الناتج كالتالي:

مثال ١٦:

سنحاول الآن إدخال ثلاث قيم عددية ، ومن ثم طباعتها:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int a;
float x;
double z;
cin>>a>>x>>z; // input all data
cout<<endl;
cout<<"\n print the following items";
cout<<endl;
cout<<a<<"\n"<<x<<"\n"<<z; // output all data
cout<<"\n this is the end"<<endl;
return0;
}
```

لتكن 10 (← int a;)

لتكن 6.3 (← float x;)

لتكن 2001.5 (← double z;)

نلاحظ في السؤال أعلاه أننا قمنا بإدخال ثلاث قيم مختلفة من حيث النوع وأيضا قمن باستخدام التعليقات وهي لا تؤثر في البرنامج فقط للتوضيح وهي التعليق:

// input all data

// output all data

لنوضح للمستخدم أين الإدخال والإخراج ..

وسوف يكون الناتج كالتالي:

print the following items
10
6.3
2001.5
this is the end

# جمل التحكم والشرط والتكرار
# ProgramControl,Conditional&Loop Statements

## introduction مقدمة

قد نحتاج أن ننتقل من سطر إلى آخر في برنامج C++ ، وقد نحتاج أن نقوم بتنفيذ بعض الجمل عند تحقق بعض الشروط ، وقد نحتاج أن نكرر عملية من عمليات الإدخال أو الإخراج أو الحساب عددا من المرات ، وقد نحتاج أن نبني شبكة من توزيع الأوامر على عدد من سطور البرنامج ، حسب مقتضيات بعض الأحوال ، وحسبما تقتضيه طبيعة المسألة ، في هذه الحالات: نحتاج أن نتعلم أساليب الشرط ، وأساليب التكرار ، وكيفية التحكم في مسار البرنامج ، وتعد أساليب الشرط والتحكم والتكرار بمثابة القلب في جسم لغات البرمجة ، وبدونها لا يمكن تنظيم آي برنامج.

وتوفر لغة C++ للمبرمج عددا من الأساليب والدوال الفعالة ، المتعلقة بهذا الشأن ، وتمتاز هذه الأساليب بأنها أساليب بنائية أو بنيوية structured آي يمكن تنظيم عمليات التحكم والتكرار فيها ، بأسلوب ذاتي من بداية العمليات وحتى نهايتها دون تدخل من المبرمج أثناء هذه ا لعمليات ، للأشراف على التوجيه والتخطيط لكل خطوه من خطوات البرنامج ، ويعرف بعض الخبراء والمختصين البرمجة البنيوية: أنها البرمجة التي لا تستعمل جملة الانتقال GOTO ، لتوجيه البرنامج في كل خطوة ، ومع ذلك فان لغة C++ ، توفر جملة الانتقال هذه ، لكنها لا تستعمل إلا للضرورة.

وحيث أن جواب الشرط أما أن يكون صوابا true أو زائفا false فان لغة C++ ، تعطي الحالة الصائبة قيمة عددية تختلف عن الصفر ، وتعطي قيمة صفر للحالة الزائفة ( عند عدم تحقق الشرط أو الشروط ) ولذا فان لغة C++ توفر مرونة كبيرة في استخدام عدد كبير من الدوال ، وفي توجيه البرنامج بطريقة فعالة وفائقة.

# الجمل الشرطية

تتعامل لغة ++C مع ثلاثة أنواع من جمل الشرط وهي:

١- جملة إذا الشرطية وأخواتها if statements

٢- جملة التوزيع switch statement

٣- جملة أداة الشرط ?

## جملة الشرط إذا وأخواتها if statements

- جملة الشرط إذا وأخواتها if statements

تأخذ هذه الجملة الشكل العام التالي:

```
If (condition) statement1;
```

تقوم جملة إذا الشرطية هنا ، بنقل تسلسل تنفيذ البرنامج إلى الجملة (أو سلسلة الجملة) statement1 عندما يتحقق الشرط (أو الشروط) condition وتكون نتيجته true ، وإذا لم يتحقق الشرط ، أي تكون النتيجة false ، فان التنفيذ ينتقل فورا إلى الجملة ( أو سلسلة الجمل) statment2 ويعد استعمال else في ++C اختياريا ، أي يمكن حذفها دون أن تتأثر الجملة الشرطية تركيبا واداء ويكون شكلها العام على النحو التالي:

```
If (condition) statement1;
Else statment2;
```

وفي هذه الحالة ستنفذ الجملة statement1 أن تحقق الشرط condition وإلا فان التنفيذ ينتقل إلى سطر ++C التالي لجملة if .

الصيغة الأولى

وتأخذ البنية العاملة If (condition) statement1

مثال ١:

اكتب برنامجا بلغة ++C لإظهار العبارة x is positive على شاشة العرض؟

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int x=5;
if (x>0)
cout<<x<<" Saudi";
return0;
}
```

في هذا المثال ستظهر الجملة Saudi على الشاشة لان الشرط (x>0) متحقق فالخمسة بالطبع اكبر من العدد صفر فالنتيجة كالتالي:

Saudi

مثال ٢:

اكتب برنامج ++C التالي ليحسب القيمة المطلقة لـY المعرفة على النحو التالي:

$$Y=|x|=\begin{cases} x;x>=0 \\ -x;x<0 \end{cases}$$

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int x;
cin>>x;
if (x>=0) cout<<x;
else cout<<-x;
return0;
}
```

نقوم بإدخال العدد 10

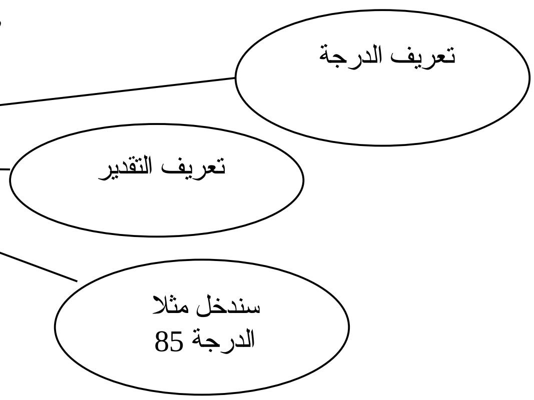
في المثال أعلاه سوف ندخل الرقم 10 لتكون النتيجة :

10

مثال٣:

قم بإنشاء برنامج لإدخال علامة طالب فإذا كانت العلامة اكبر أو تساوي 90 فالتقدير (A) أما إذا كانت اكبر أو تساوي 80 فالتقدير (B ) أما إذا كانت اكبر أو تساوي 70 فالتقدير (C) أما إذا كانت اكبر أو تساوي 60 فالتقدير (D )أما إذا كانت اكبر أو تساوي 50 فالتقدير (E ) ما عدا ذلك فالتقدير(F)؟

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int mark;
char grade;
cin>>mark;
if (mark>=90)
grade='a';
else
if (mark>=80)
grade='b';
else
if (mark>=70)
grade='c';
else
if (mark>=60)
grade='d';
else
if (mark>=50)
grade='e';
else
grade='f';
cout<<grade;
return0;
}
```




في المثال أعلاه قمنا بتعريف المتغير mark بأنه عدد صحيح ثم قمنا بعد ذلك بتعريف المتغير grade بأنه قيمة نصيه وهو التقدير.

طبعا قمنا بإدخال الدرجة وهي 85 سوف تكون العلامة كالتالي:

B

الصيغة الثانية

وتأخذ البنية العاملة لجملة إذا .... وإلا(if..else) الشكل العام التالي:

```
If (condition)
{
statmenet1;
}
else
{
statmenet1;
}
```

مثال ٤:

سوف نطبق المثال السابق (3) لكن بالشكل (if..else) أعلاه كما يلي:

```
#include  “ stdafx.h”
#include “ iostream.h”
main ()
{
int mark;
char grade;
cin>>mark;
if ( mark>=90){      <-- إذا الأولى وجوابها
grade='A';
{
else
if (mark>=80){       <-- إذا الثانية وجوابها
grade='B';
{
else
if (mark>=70){       <-- إذا الثالثة وجوابها
grade='C';
{
else
if (mark>=60){       <-- إذا الرابعة وجوابها
grade='D';
{
else
if (mark>=50){       <-- وإلا فالنتيجة الباقية هى الخامسة
grade='E';
}
cout<<grade;         <-- لطباعة التقدير
return0;
}
```

مثال٥:

اكتب برنامجا لإدخال طولك وطول زميلك ، وإذا كان طولك اكبر من طول زميلك اطبع طولك ، واحسب معدل الأطوال ، ثم اطبعه وألا اطبع طول زميلك ، واطبع ضعف الطول ونصف الطول؟

الحل/

سنرمز لطولك t1 وسنرمز لطول زميلك t2

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int t1,t2;
cin>>t1>>t2;
if (t1>t2)
{
cout<<t1;
cout<<(t1+t2)/2;
}
else
{
cout<<t2;
cout<<t2*2;
cout<<t2/2;
}
return0;
}
```

أدخل الأطوال (cin>>t1>>t2;)

طباعة معدل الأطوال (cout<<(t1+t2)/2;)

وإلا (else)

لطباعة ضعف الطول (cout<<t2*2;)

لطباعة نصف الطول (cout<<t2/2;)

# جملة التوزيع switch statement

تأخذ جملة Switch الشكل العام التالي في لغة c++ :

```
Switch (expression)
{
Case constant1: statments1; break;
Case constant2: statments2; break;
.............................................
.............................................
.............................................
Case constantn: statmentsn; break;
Default; last statements;
}
```

تعبير قيمته صحيحة (Switch (expression))

جملة أو جمل تنفذ عندما ساوي التعبير قيمة constant1

جملة أو جمل تنفذ عندما ساوي التعبير قيمة constant2

يتم تنفيذها عند عدم تنفيذ جميع الحالات الواردة أعلاه في بنية Switch

مثال٦:

```
#include  " stdafx.h"
#include " iostream.h"
void main()
{
int s1;
s1=2;
switch (s1)
{
case 2 :cout<<"y";
          break;
case 3: cout<<"x";
          break;
case 4: cout<<"m";
          break;
default: cout<<"w";
}
 }
```

والنتيجة:

y

## جملة أداة الشرط ?

وهـي أداة سـريعة مكافئة لبنـية إذا ... وإلا ، وقـد مـر معـنا كيفـية اسـتعمالها فـي أول الكتاب وسوف نورد هنا صورتها العامة:

```
Variable=(condition)? Result:result2;
```

ومعـناها: انــه يــتم تنفـيذ النتـيجة الأولــى result1 عـندما يكـون جـواب الشــرط condition مـتحققا (true) ، وإلا فيـتم تنفـيذ النتـيجة الثانـية result2 عـندما يكـون جواب الشرط (false) .

مثال٧:

```
#include  " stdafx.h"
#include " iostream.h"
void main()
{
int a,b;
a=5;
if (a>1) b=10;
else
b=20;
cout<<b;
}
```

ومعناها أن b تـأخذ القيمة 10 إذا تحقق الشـرط a>1 وتـأخذ القيمة 20 إذا لـم يتحقق الشرط نفسه .

والنتيجة:

10

# التكرار وحلقات التكرار Repetition and Loops

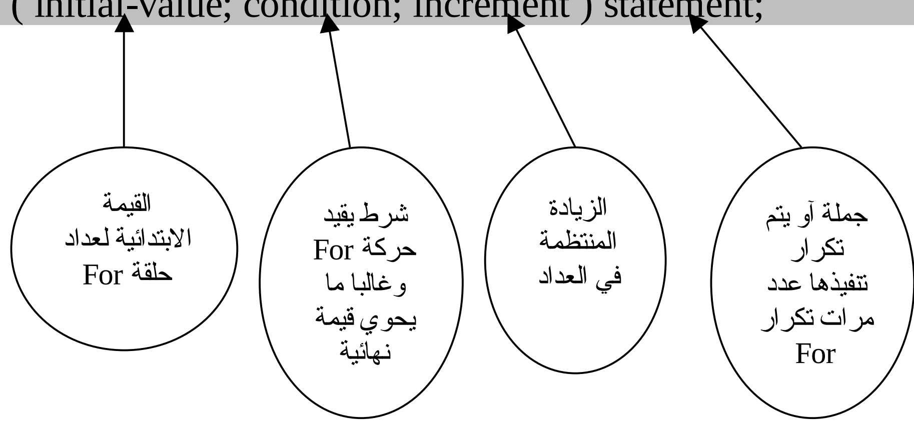
توفر لغة ++C ، كسائر لغات البرمجة ، عددا من أساليب التكرار المشروط ، وغير المشروط ومن هذه الأساليب:

# أسلوب التكرار باستعمال حلقة For

يمتلك أسلوب التكرار باستعمال for قوة ومرونة ، لا تتوفران في غيرها من اللغات.

الصيغة العامة الأولى

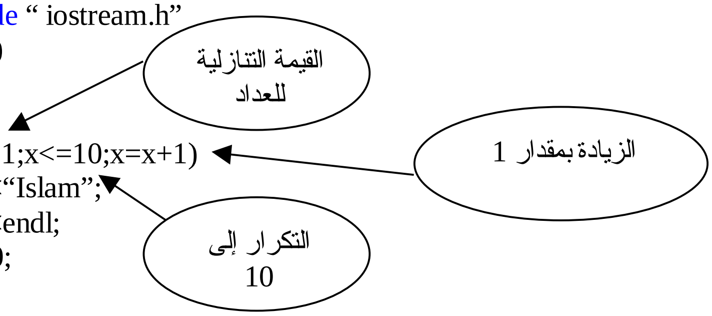
For ( initial-value; condition; increment ) statement;




تقوم جملة For هنا مبتدئة بقيمة العداد الابتدائية بتنفيذ الجملة statement1أول مرة ، وفي المرة التالية تزداد القيمة الابتدائية للعداد بمقدار الزيادة ثم تنفذ جملة statement1 مرة ثانية .. وهكذا حتى يستكمل الشرطconditionأمر إنهاء عمليات التكرار والخروج من حلقة For ، والأمثلة التالي توضح كيفية استعمال حلقات التكرار بجملة For:

مثال٨:

```
#include  “ stdafx.h”
#include “ iostream.h”
main ()
{
char x;
for (x=1;x<=10;x=x+1)
cout<<“Islam”;
cout<<endl;
return0;
}
```




والنتيجة كالتالي:

Islam Islam Islam Islam Islam Islam Islam Islam Islam Islam

نلاحظ هنا انه تم تكرار كلمة Islam 10 مرات بداية من القيمة 1 Ù 10

مثال ٩:

اكتب برنامجا لطباعة قيمة العداد من 1 إلى 10؟

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int a;
for (a=1;a<=10;++a)
cout<<a<<endl;
return0;
}
```

وتكون نتائج الطباعة على الشاشة هكذا:

1 ← قيمة a الابتدائية
2
3
4
5
6
7
8
9
10 ← قيمة a النهائية

مثال ١٠:

اكتب برنامجا لطباعة الأعداد الفردية من 1 إلى 15؟

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int a;
for (a=1;a<=15;a=a+2)
cout<<a<<endl;
return0;
}
```

ومن الملاحظ أننا جعلنا قيمة الزيادة 2 وليس 1 لانه طلب أعداد فردية بداية بالقيمة 1 وحتى 15

و النتيجة كالتالي:

1
3
5
7
9
11
13

الصيغة العامة الثانية

```
For ( initial-value; condition; increment )
{
statement;
}
```

جملة أو اكثر

شاهد الأمثلة التالية لنتعرف اكثر على الصيغة أعلاه:

مثال ١١:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{ int x,y,z;
y=-4;
for(x=1;x>y;x=x-2)
}
z=x;
cout<<x<<endl;
{
return0;
}
```

اكثر من جملة بين
القطعتين Block

والناتج سوف يكون التالي:

```
1
-1
-3
```

مثال ١٢:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{ int a,b,c,x;
a=6;
b=1;
c=3;
for (x=11;(a*c)>(x*b);x--)
{
x=x+3;
c=c-2;
cout<<x<<"*";
}
return0;
}
```

القيمة الابتدائية

اكثر من جملة بين
القطعتين Block

14*

مثال ١٣:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{ int a,c;
a=1;
c=3;
for (a=c;c;)
{
c--;
cout<<c<<endl;
}
return0;
}
```

تنقص من قيمة
C قيمة 1

والناتج:

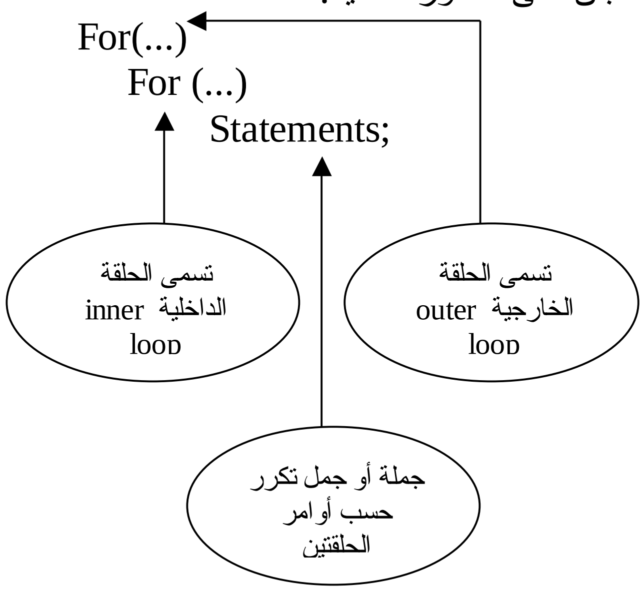
2
1
0

# حلقات التكرار المتداخلة Nested (Multiple) for Loops

تأخذ صيغة حلقات التكرار المتداخلة الشكل العام التالي:

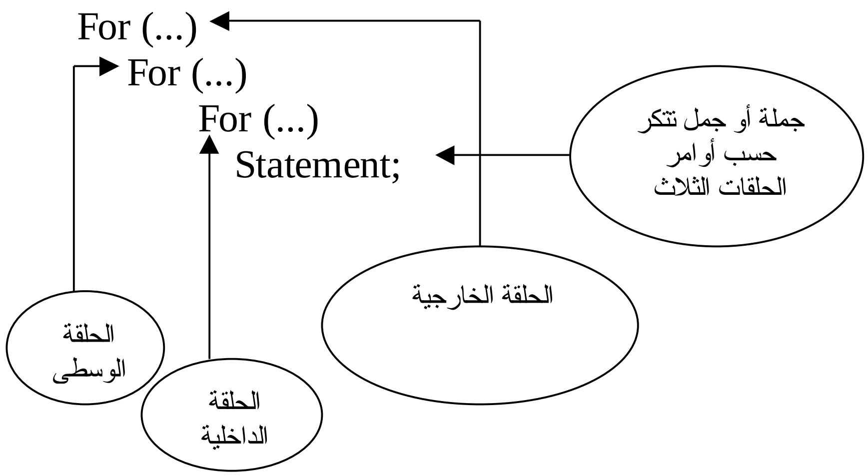
```
For (...)
    For (...)
          For (...)
           ..............................
                     ..............................
                                Statements;
```

فلو أخذنا حالة حلقتين متداخلتين فانهما تكتبان على الصورة التالية:




وتكون في هذه الحالة الجملة (أو الجمل) جزءا مكررا مرتبطا بالحلقة الداخلية ، والحلقة الخارجية تتكرر حسب أوامر الحلقة الخارجية وهكذا ...

وفي حالة الثلاث حلقات المتداخلة ، فإنها تكتب على الصورة التالية:




مثال ١٤:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int i,j;
for (i=1;i<=3;++i)
for (j=1;j<=4;++j)
cout<<i*j;
return0;
}
```

الحلقة الخارجية تتكرر 3 مرات

الحلقة الداخلية تتكرر 4 مرات لكل قيمة من قيم I أي مجموع 12 مرة

لاحظ هنا أن الحلقة الداخلية تتكرر ٤ مرات لكل قيمة من قيم I ، عداد الحلقة الخارجية ، وكذلك جملة cout<< ، وبما أن I ، تأخذ 3 قيم فان الحلقة الداخلية تتكرر 12 مرة ، أما الحلقة الخارجية فتكرر نفسها بنفسها فتتكرر 12 مرة فقط.

والناتج:

1234246836912

مثال١٥:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int i,j;
for (i=2;i<=20;i+=2)
for (j=30;j>=3;j-=3)
cout<<i<<j<<endl;
return0;
}
```

الحلقة الخارجية يتغير عدادها
بين 2 و 20 بزيادة منظمة
قدرها 2

الحلقة ا لداخلية يتغير عدادها بين
30 و 3 بزيادة منتظمة قدرها-3

والناتج سيكون كبير لذلك سأعطيكم جواب الحل للسطر الأول والسطر الأخير وما بينهما لكم.

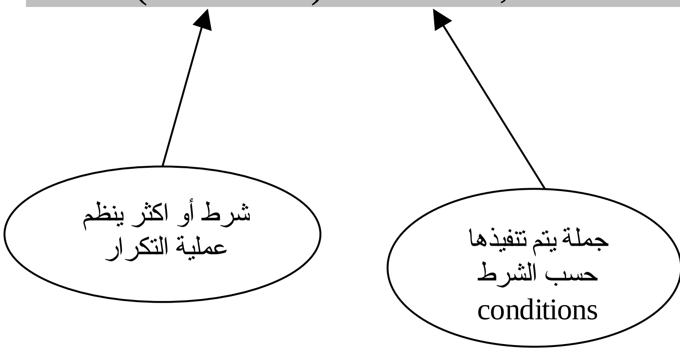
السطر الأول: 1612

السطر الأخير : 203

# أسلوب التكرار باستعمال حلقة While & Do

أسلوب التكرار باستعمال حلقة while أسلوب آخر يماثل أسلوب حلقة for ، مع بعض الاختلافات البسيطة ، وهو أسلوب يثرى لغة C++ ، ويزدها قوة ومرونة ، والشكل العام لهذا الأسلوب:

While (conditions) statements;



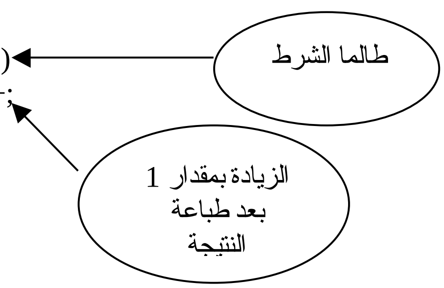

ومعنى حلقة التكرار while هو الأتي:

آي ما دام الشرط ( أو الشروط ) متحققا (وجوابه true) ، فيتم تكرار تنفيذ الجملة أو الجمل (statements) ، وينتقل تسلسل تنفيذ البرنامج إلى الجملة التي تلي حلقة while .

والأمثلة التالية توضح ذلك:

مثال ١٦:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{ int a;
a=1;
while (a<3)
cout<<a++;
return0;
}
```




سوف يكون:

1 2

مثال١٧:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{ int a;
a=1;
do
{
cout<<"*";
a++;
}
while (a<=4)
return0;
}
```

افعل ← do

الزيادة بمقدار القيمة 1 ← a++;

طالما a اقل أو يساوي القيمة 4 ارجع التكرار ← while (a<=4)

والناتج

****

مثال١٨:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{ int x,y;
x=2;
y=-2;
do
{
cout<<x<<"-"<<endl;
x--;
y++;
}
else
cout<<"finish";
while(x!=y)
return0;
}
```

قطعتين أو بلوك لوضع الجمل بينهما ← { }

طالما x لا تساوي y تابع التكرار أو الدوران ← while(x!=y)

2--21--1

مثال ١٩:

اكتب برنامجا يطلب من المستخدم إدخال قيمة عددية ، وطالما أن القيمة المدخلة + يطبع * على سطر جديد؟

الحل/

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{ int a;
cout<<"please enter a number";
cin>>a;
while (a>=0)
{
cout<<"*"<<endl;
cout<<" please enter a number";
cin>>a;
}
return0;
}
```

ندخل الرقم وليكن 10

طالما أن الرقم المدخل اكبر أو يساوي 0

إذا تحقق الشرط اطبع

طبعا قمنا بإدخال الرقم 10 والنتيجة ستكون:

*

لان العدد 10 عدد موجب لكن حاول أن تدخل عدد سالب فلن يطبع لك شي لعدم تحقق الشرط ، ولا ننسى أخواني أننا وضعنا الجمل أو الجملة الخاصة بالطباعة بين القطع Block {} لانه وجد اكثر من جملة لذلك يجب وضع القطع لكن عند عدم وجود اكثر من سطر أو جمله كمثال (١٢) فلا يجوز وضعها.

# حلقات While المتداخلة Nested While Loops

تشـبة حلقـات While المـتداخلة حلقـات for المـتداخلة ، فمـثلا خـذ حلقتـي الـتكرار المتداخلتين التاليتين:

مثال ٢٠:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int i=j=1;
while (i<=3)
{
while(j<=4)
{
cout<<"/n"<<i*j;
++j;
}
++i;
j=1;
}
return0;
}
```

- while (i<=3) ← حلقات تكرار 3 مرات
- while(j<=4) ← الحلقة الداخلية تتكرر 12 مرة
- ++i; ← زيادة منتظمة في عدادI بمقدار 1
- j=1; ← إعادة تعريف قيمة j الابتدائية عند بداية حلقة j

# جملة الإيقاف Break

من الاسم نستطيع أن نلاحظ أن وظيفة Break هي إيقاف بنية أو حلقة تكرار عند تحقق شرط أو شروط معينة ، وعند تنفيذها يتم القفز إلى سلسلة الجمل التالية للبنية أو حلقة التكرار ، وتستعمل Break أيضا في إيقاف حلقة التكرار لانهائي ، أو الخروج منها إلى الجمل التي تليها وكما في المثال التالي:

مثال ٢١:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int i;
for (i=1;i<100;++i)
{
cout<<i;
if (i= =10) break;
}
return0;
}
```

يوقف تنفيذ هذه الجملة حلقات التكرار عندما يصبح i=10

وطبعا سيقوم بتنفيذ البرنامج حتى العدد 10

والنتائج:

12345678910

مثال ٢٢:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int number;
for (number=1; number<=100;++ number)
{
if (number%2) // for ood values
break;
else if (number%4)
break;
else if (number%6)
break;
else
cout<< number<<endl;
}
return0;
```

# جملة الاستمرار continue

تعمل جملة الاستمرار continue على تجاوز تنفيذ بقية الجمل في التكرار خلال الدورة الحالية والانتقال إلى الدورة الثانية:

مثال٢٣:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int x,n;
do
{
cin>>x>>n;
if (n<1) continue;
cout<<x;
--n;
}
while (n<1);
return0;
}
```

تعمل على تجاوز تنفيذ
الجملتين التاليتين
وتبداء دورة جديدة إذا
تحقق الشرط

مثال ٢٤:

تطبع جميع الأرقام من 1 إلى 100 ما عدا الأرقام التي تقسم على 2 و 4 و6 بدون باق:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int number;
for (number=1;number<=100;++number)
{
if (number%2)
continue;
else if (number%4)
continue ;
else if (number%6)
continue;
else
cout<<number<<endl;
}
return0;
}
```

الناتج:

12
24
36
48
60
72
84
96

## جملة الخروج exit()

تعمل هذه الدالة على إيقاف ( أو الخروج من ) البرنامج في مكان منه ، وتشبة end في لغة بيسك ، وتكون قيمة الدالة صفرا exit(0) عندما يتم الخروج من البرنامج بنجاح وألا فان قيمة الدالة تكون exit(1) وتوقف البرنامج عند وجود خطأ أو نحو ذلك ، وفي هذه الحالة ، وتلك يعود البرنامج تنفيذه إلى نظام التشغيل operating system .

مثال٢٥:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
chat ma;
cin>>ma;
if ( ma != 'A') exit(1);
cout<<"\n"<<ma;
return0;
}
```

## جملة الانتقال goto

من المعروف أن معظم لغات البنيوية الحديثة ، تحرص ، في غالب الأحيان ألا تستعمل جملة goto من اجل التأكيد على المبرمج ، أن يتعلم برامجه بطريقة بنيوية structured ذاتية المداخل والمخارج ، والعمليات ، دون تدخل من المبرمج بقوله : اذهب goto من هنا ، أو اذهب من هناك آي أن البرنامج في هذه الحالة يعتمد على نفسه .

مثال٢٦:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int x;
cin>>x;
if (x<10) goto negative;
negative: cout<<"value is under 10"<<endl;
return0;
}
```

في هذا السؤال يطلب من المستخدم إدخال قيمة فإذا كانت القيمة اقل من 10 حسب الشرط فانه يعرض لك الرسالة value is under 10 .
أما إذا كانت اكبر من العدد 10 فانه يطبع العدد مباشرة من دون الذهاب للسطر الأخير لتحقق الشرط .

# المتغيرات المرقمة والمصفوفات
# Arrays and Matrices

## مقدمة introduction

أن طرق التعامل مع أسماء المتغيرات والثوابت العددية والرمزية ، التي وردت في الفصول السابقة ، تعد صالحة للتعامل مع عدد محدود من هذه الثوابت والمتغيرات ، سواء في عمليات الإدخال والإخراج أو في العمليات الحسابية والمنطقية ، وعندما يصبح عدد المتغيرات كبيرا جدا ، تصبح تلك الطرق غير عملية ، فمثلا لو أردنا إدخال مائة قيمة للمتغيرات -x2,x1 .... إلى x100 ، فكم الحيز المطلوب من البرنامج لعمليات الإدخال والإخراج والعمليات الحسابية والمنطقية لهذه المتغيرات ؟ هذا من جهة ، ومن جهة أخرى : فأننا نوفر مخزنا خاصا لكل متغير نتعامل معه ، أثناء تنفيذ البرنامج ، ولذلك لحفظ قيمته في مخون ، ومن ثم لاستعمال قيمته في عمليات أخرى تالية ، ومن ناحية ثالثة ، فان من الصعوبة بمكان ، بل من المستحيل استعمال اسم المتغير العددي أو الرمزي كمصفوفة ذات بعدين ، وثلاثة أبعاد ...الخ

للأسباب الثلاثة الواردة أعلاه ، جاءت فكرة استعمال متغير جماعي يضم تحت اسمه عددا من العناصر يسمى بالمتغير الرقمي subscripted variable ، ويتم ترقميه بين قوسين مربعين [ ] يوضع بينهما قيمة العداد المرقم subscript ، وقد نسمية الدليل index أحيانا ، ويمكننا تشبيه المتغير المرقم بقسم الهاتف لمؤسسة ما ، فهو مقسم واحد ، تنظم تحته عدد من الأرقام الفرعية للموظفين وكل رقم من هذه الأرقام مستقل ومتميز عن الأرقام الفرعية الأخرى ، وله مخزن خاص في الذاكرة ، الآن انه كغيره من الأرقام الفرعية تابع للرقم العام لمقسم المؤسسة ، كما يمكن تشبيه المتغير المرقم بالجيش الذي يعامل كاسم متغير واحد ، لكن يضم عددا كبيرا من العناصر ، فمثلا العناصر التالية :(من اليمين إلى اليسار):

A[n] ...a[2], a[1], a[0]

تابع للمتغير الجماعي [ ]a

وكل عنصر من هذه العناصر له عنوان في الذاكر address ، فالعنوان الأول يكون للعنصر الأول والثاني للثاني والثالث للثالث ... وهكذا.

ويستعمل المتغير الجماعي [المرقم] أو المصفوفة ، في لغة ++c وغيرها ، حجز جماعي مسبق في الذاكرة لجميع عناصره ، فلو كان يتبعه خمسون عنصرا ، فانه يحجز له 50 مخزنا ، على الأقل في الذاكرة .

من الفوائد المهمة للمتغيرات المرقمة والمصفوفات : هو استعمالها في الترتيب التصاعدي والتنازلي للعناصر والقيم المختلفة ، وعمليات ترتيب الأسماء الأبجدي

النصوص الرمزية ، وفي عمليات ضرب المصفوفات ، وإيجاد معكوس المصفوفة وعملياتها الأخرى ، وفي التحليل العددي ... الخ.

المتغير المرقم ( المصفوفة ) ذو البعد الواحد one-dimensional Array

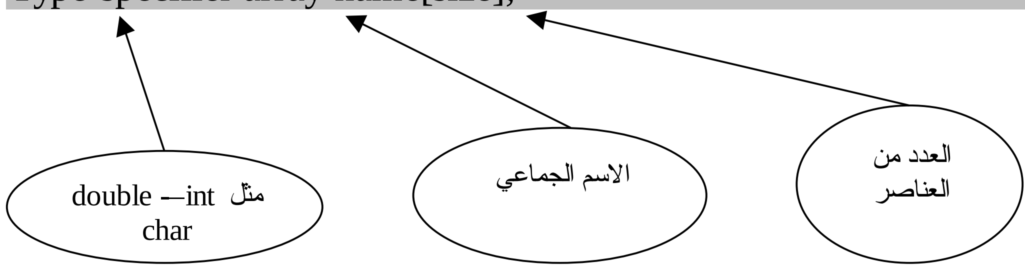
المتغير المرقم ذو البعد الواحد هو مصفوفة ذات بعد واحد أو متجه (vector) ويمثل في الجبر على النحو الأفقي [a1 a2 ...a3]

أو العمودي

$$\begin{pmatrix} A1 \\ A2 \\ : \\ : \\ : \\ a3 \end{pmatrix}$$

ويأخذ المرقم المتغير في c++ الشكل العام التالي:

Type-specifier array-name[size];




ويبدأ العداد المرقم عادة من الصفر ، آي أن العنصر الأول من المصفوفة a[] هو a[0] والثاني a[1] ... وهكذا فمثلا المصفوفة التالية:

```
Int a[20];
```

اسمها a ، وقد حجز لها 20 موقعا لعشرين عنصرا من النوع الصحيح .

والمصفوفة التالية:

```
Char name[15];
```

مصفوفة رمزية ، اسمها name يحجز لها خمسة عشر عنصرا من النوع الرمزي لها .

وهكذا...

مثال ١:
مثال على عملية إدخال ذاتي لقيم عناصر متغير مرقم ( مصفوفة ) ذي بعد واحد

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int a[20];
int I;
for (I=0;I<20;++I)
a[I]=I+1;
return0;
}
```

عداد مرقم المصفوفة

في هذه الحالة يتم إدخال عشرين عنصرا من عناصر المصفوفة a
A[0]=1 عندما يكون I=0
A[1]=2 عندما يكون I=1
…
…
…
a[19]=20 عندما يكون I=19

مثال ٢:
مثال على عمليات إدخال ، وحساب ، وعمليات طباعة عناصر مصفوفة:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int x[5], y[5];
int I;
for (I=0;I<5;++I)
{
x[I]=I;
y[I]=I*I;
cout<<endl<<x[I]<<y[I];
}
return0;
}
```

وستكون قيم النتائج على النحو التالي:

```
0    0
1    1
2    4
3    9
4   16
```

# إعطـاء قيمة أولية للمصفوفة ذات البعد الواحد Array Initialization

مثال على إدخال عدة عناصر من مصفوفة الدرجات grade[]

Int grade[5]={80,90,54,50,95}

ومثال على إدخال قيم عناصر المصفوفة الرمزية name[]

Char name[4]=“nor”

لاحـظ أن المتغير المـرقم name[] مكـون مـن أربعـة عناصـر بيـنما تـم إعطـاؤه ثلاثـة عناصر فقط والسبب أن العنصر الرابع بالنسبة إلى المعطيات الرمزية يكون خاليا.

مثال٣:

```
#include  “ stdafx.h”
#include “ iostream.h”
main ()
{
int a[6]={40,60,50,70,80,90}
int I;
for(I=0;I<6;I++)
cout<<a[I]<<endl;
return0;
}
```

تم إعطاء القيم من قبل المستخدم مسبقا هنا

والناتج طبعا سيكون كالتالي:

40
60
50
70
80
90

مثال ٤:

قم بكتابة برنامج يقوم بإيجاد مجموع ، ومعدل علامات الطالب في 5 مواد وهذه العلامات كالتالي:

87,67,81,90,55

```
#include  " stdafx.h"
#include " iostream.h"
int m,i;
main ()
{
int a[5]={87,67,81,90,55}
int s=0;
for(i=0;i<5;i++)
s=s+m[a];
float avg=s/5;
cout<<avg<<endl;<<s<<endl;
return0;
}
```

قيمة المعدل لجميع العلامات

والناتج سيكون كالتالي:

87
735

المعدل 87
والمجموع 735

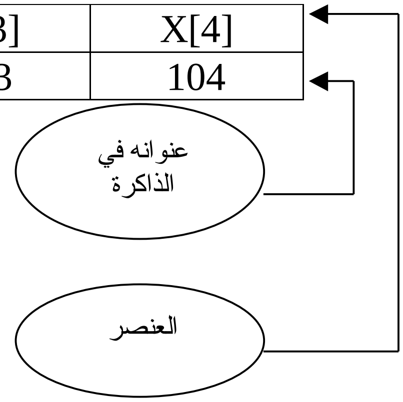
# عـنوان عناصـر المصـفوفة فـي الذاكـرة Addressing Array Elements in Memory

ذكرنا من قبل أن آي متغير أو عنصر من متغير ذاتي مرقم ، يحتل موقعا من الذاكرة يستعمل عادة مؤشرا لكل متغير أو عنصر ، ليكون دليلا على استعمال هذه المتغيرات والعناصر بسهولة ويسر ، والمثال التالي يوضح هذه العملية بالنسبة للمصفوفة ذات بعد واحد.

Int x[5];

يمكن تمثيل عناصر المصفوفة x المعلن عنها ، مع عناوينها بالشكل التوضيحي التالي ( من اليسار إلى اليمين)

| الأول | الثاني | الثالث | الرابع | الخامس |
|---|---|---|---|---|
| X[0] | X[1] | X[2] | X[3] | X[4] |
| 100 | 101 | 102 | 103 | 104 |



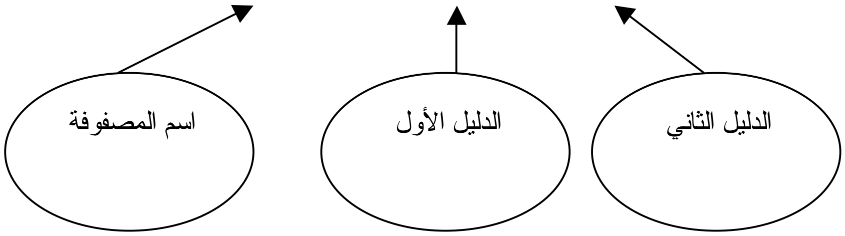

إذا فرضنا أن عنوان موقع العنصر الأول x[0] في الذاكرة هو 100 ، فان عناوين العناصر الأخرى تكون على التوالي 101 102 103 104

يمكن تشبيه العلاقة بين قيمة العنصر ، وعنوانه ، بالعلاقة بين علامة طالب ، ورقمه الجامعي ، إذ علامته هي قيمة نشطه كعنصر ، ليس لها علاقة برقم مقعده الجامعي.

# المصفوفة ذات البعدين Two-Dimensional Arrays

تشبة المصفوفة ذات البعدين في طريقة تعاملها ، المصفوفة ذات البعد الواحد إلا أن لها عدادين( index2) دليلين أو مرقمين إحداهما عداد للصفوف ، والأخر عداد للأعمدة ويأخذ الإعلان عن المصفوفة الشكل العام التالي:

```
Type-specifier arraey_name [index 1][index 2];
```




فمثلا المصفوفة :

```
Int x[2][3];
```

وهي مصفوفة صحيحة العناصر int أبعادها هي عدد الصفوف=2 ، وعدد الأعمدة =3

لاحظ أن عدد الصفوف يوضع بين قوسين وحده ، وكذلك عداد الأعمدة .

مثال٥:

شاهد هذا المثال الذي يستخدم 5 طلاب و 3 علامات:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int m[5][3];
int I,j;
for(I=0;I<5;I++)
for(j=0;j>3;j++)
cin>>m[I][j];
return0;
}
```

وبالنسبة لعناوين العناصر المصفوفة متعددة الأبعاد في الذاكرة ، لا يختلف عما ذكرنا بالنسبة للمصفوفات ذات البعد الواحد ، ولذلك لو فرضنا ، في المثال السابق أن العنصر x[0,0] كان عنوانه 100 مثلا فان عناوين العناصر التالية: حسب ترتيبها المذكور أعلاه هي 100-101-102 لعناصر الصف الأول 103-104-105 لعناصر الصف الثاني.

# الدوال
# Functions

## مقدمة Introduction

تعرف الدالـة ( الاقتران ) علـى أنها : جملـة أو مجموعـة جمل أو تعليمات ، ذات كيان خـاص ، تقـوم بعملـية أو مجموعـة عملـيات ، سـواء عملـيات إدخـال أو إخـراج أو عمليات حسابية أو منطقية ، وتحتل الدالـة موقعا من البرنامج ، آي أنها جزء منه ، أو يمكن القول أن برنامج ++c ، يتكون من مجموعة من الدوال.

ومن فوائد الدوال التالي:

١- تساعد الـدوال المخـزنة فـي ذاكـرة الحاسـب علـى اختصـار الـبرنامج إذ يكـتفى باستعادتها باسمها فقط لتقوم بالعمل المطلوب.
٢- تساعد الـدوال المخـزنة فـي مكتبة الحاسب ، أو التي يكتبها المبرمج على تلافي عمليات التكرار في خطوات البرنامج التي تتطلب عملا طويلا وشاقا.
٣- تساعد الدوال الجاهزة على تسهيل عملية البرمجة نفسها.
٤- توفر مساحة من الذاكرة المطلوبة.
٥- اختصار عمليات زمن البرمجة وتنفيذ البرنامج بأسرع وقت ممكن.

وللتدليل على أهمية الدوال في برمجة ++c خذ المثال التالي:
لو أردنا كتابة خوارزمية لخطوات صنع كأس من الشاي فأننا نكتب ما يأتي:

١- ضع الماء في غلاية الشاي.
٢- سخن الماء حتى يغلي.
٣- أضف شايا إلى الماء.
٤- أضف سكرا إليه.
٥- أطفئ النار.
٦- صب شايا في كأس.

افـرض الآن أنـنا نـود طلـب كـأس مـن الشـاي مـن مقهـى مجـاور : أن خطـوات الخوارزمية التي نحتاجها الآن هي خطوه واحده فقط وهي:

١- استدع كأس من الشاي.

تخيل الآن كم وفرنا من الخطوات لو استعملنا الـدوال الجاهـزة ( أو التـي يجهـزها المبرمج من قبل ) بدلا من خطواتها التفصيلية وبخاصة في برنامج يتطلب حسابات وعمليات كثيرة وكم يكون البرنامج سهلا وواضحا وقتذاك.

وتأخذ الدالة الشكل العام التالي:

```
Type-specified function-name (formal parameters;
{
function body
}
```

وقد ذكرنا من قبل أن الدالة قد تعتمد على متغير أو اكثر ، وقد لا تعتمد على آي متغير ، وفي كلا الحالتين ، يستعمل بعد اسم الدالة قوسين () سواء كان بينهما متغيرات أم لا .

مثال ١:

```
#include  " stdafx.h"
#include " iostream.h"
max1()
{
cout<<"hello";
}
void main()
{
max1 ();
max1();
max1(); max1();
}
```

والناتج:

hello hello hello hello

طبعا للعلم أعزائي أننا في هذا الفصل الدوال نلاحظ أن بداية قراءة المترجم للبرنامج لا تبدأ من أول البرنامج كالمعتاد فالقراءة تبدأ من الأسفل آي أنها تبدا بالماين main سواء كان في الوسط أو الأسفل فأنها تقراء أولا الـmain ثم تبحث ما داخله وتبدا بالبحث عن معنى الكلمة max1() في الدالة max1() في الأعلى لتجد أن هناك جملة طباعة وهكذا تتكرر حتى يتم تعريف ما بداخل الـmain .

# تطبيقات على الدوال

مثال٢:

شاهد هذا البرنامج وتتبعه أولا بالـmain وانظر للنتائج:

```
#include  " stdafx.h"
#include " iostream.h"
int x,y;
void max()
{
x=x+y;
}
void fax()
{
max();
max();
{
void main()
{
y=10; x=0;
max();
fax();
cout<<x<<y;
}
```

أعزائي سأشرح النتائج قبل إظهارها للتسهيل عليكم في الأمثلة القادمة:

لنتعبر أن هناك ثلاث كواكب:

كواكب main الرئيسي

كوكب fax

كوكب max

من المعروف أننا سوف نبدا بكوكب main لنشاهد ما بداخل نشاهد أن هناك قيمتين عدديتين x y لكنه لا يعرفه هل هي أعداد حقيقية أم صحيحه لذلك يذهب في الأعلى ليبحث عنها في أول البرنامج ليجد أنها أعداد صحيحة int ، ثم بعد ذلك يرجع للكوكب الرئيسي main ليشاهد عبارة max() فيذهب للبحث عنها في الكوكب max طبعا ليجد بداخلها أن قيمة x تساوي x+y أي أن x=0+10 لتصبح قيمة x=10 بعد ذلك يخرج من الكوكب max ويرجع للكوكب الرئيسي ليشاهد العبارة fax() فيذهب للتعرف عليها بالكوكب fax() ويشاهد بداخلها عبارة max ليذهب بذلك للكوكب max ويجمع من مرة أخرى فتصبح كالتالي:

X=10+10 وبذلك تصبح قيمة x=20 بعد ذلك يرجع للكوكب fax ليشاهد عبارة max() فيذهب للكوكب max ويجمع مرة أخرى كالتالي:

X=20+10 وبذلك تصبح قيمة x=30

ثم بعدها يرجع للكوكب الرئيسي main ليشاهد جملة الطباعة والناتج كالتالي:

30 10

مثال٣:

```
#include  " stdafx.h"
#include " iostream.h"
void x1()
{
cout<<"*";
}
void x2()
{
cout<<"\t";
}
void yaya()
{
x1();
x2();
x1();
}
void kiki()
{
cout<<"\n":
}
void main()
{
int I;
for(I=0;I<=3;I++)
{
yaya();
kiki();
}
}
```

والناتج سيكون كالتالي:

```
*         *
*         *
*         *
*         *
```

مثال ٤ :

قم بكتابة برنامج يقوم بقراءة عدد صحيح ومن ثم طباعة ما إذا كان الرقم زوجي أم فردي من خلال دالة أو اقتران؟

```
#include  " stdafx.h"
#include " iostream.h"
int x;
check()
{
if ((x%2)==0)
cout<<"even"<<endl;
else
cout<<"ood"<<endl;
}
main()
{
cin>>x;
check ();
return0;
}
```

اطبع إذا كان زوجي

اطبع إذا كان فردي

ادخل القيمة 9 مثلا

طبعا والناتج سيكون حسبما أدخلت لندخل مثلا القيمة 9.. والناتج سيكون:

ood

لان القيمة المدخلة 9 عدد فردي وليس زوجي .

مثال٥:

قم بكتابة برنامج يقوم باستخراج اكبر رقم ما بين رقمين مدخلين من قبل المستخدم وذلك من خلال دالة اسمها max ؟

```
#include  " stdafx.h"
#include " iostream.h"
int x,y;
max()
{
if (x>y)
cout<<x<<endl;
else
cout<<y<<endl;
}
main()
{
cin>>x>>y;
max ();
{
return0;
}
```

للمقارنة من الأكبر بين القيمتين

ادخل القيمتين ولتكن القيمة 10 و 20

والناتج طبعا سيكون:

20

لان العدد الذي أدخلنا 20 اكبر من العدد الذي أدخلنا 10 فقام بطباعته حسب الشرط .

# تقنية الأقراص و دوال الملفات الانتقالية
# disk Files

## مقدمة Introduction

صممت هذه الدوال للتعامل مع الملفات الانتقالية للأقراص Buffered filw system لمعالجة النصوص ، كما كان متوفرا في لغة C++ النظام الأخر غير الانتقالي unbuffered المشابهة لنظام يونيكس للإدخال والإخراج UNIX-like I/O ، وكان النظام الأخير يستعمل للتعامل مع المعطيات بنسق النظام الثنائي binary system ، ألا أن لجنة C في معهد المقاييس الأمريكي الوطني للغات البرمجة Committee ANISI-C قررت مؤخرا الاستغناء عن النظام غير الانتقالي من أنظمة التعامل مع المعطيات الثنائية ، فأوجدت البديل ، بحيث يصبح بمقدور دوال النظام الانتقالي التعامل مع كل من النصوص text والمعطيات الثنائية binary system في وقت واحد . أي أن آخر صورة (version) من C ، يتعامل بنظام واحد فقط هو نظام الملفات الانتقالية .

ونحتاج عند التعامل مع دوال هذا النظام ، أن نستعمل ملف الدليل للإدخال والإخراج stdio.h ويلخص لنا الجدول التالي اشهر هذه الدوال:

| اسم الدالة | وظيفتها |
|---|---|
| fopen() | تفتح لك ملفا |
| fclose() | تغلق لك ملفا |
| putc() | تخرج ( تطبع ) لبنة ( رمزا ) وهي مثل char |
| getc() | تدخل لبنة ( رمزا ) إلى الملف ، وهي مثل char |
| fseek() | تبحث لك عن بعض الرموز ( نص) في الملف |
| fprintf() | مثل وظيفة printf لكن للملفات |
| fscanf() | مثل وظيفة scanf لكن للملفات |
| feof() | تعطي النتيجة true عند وصول نهاية الملف |
| ferror() | تعطي النتيجة true عند حدوث خطأ |
| rewind() | ترجع الملف إلى بدايته |

**الجدول ١ ـ ٥**

# دالة فتح الملف fopen ()

مثال ١:

```
#include “stdio.h”
main()
{
FILE *f;
F=fopen(“C:\\matrix.txt”,”w”);
Fclose (f);

return0;
}
```

نكتب آي اسم للتخزين بموقع الذاكرة مثلا f

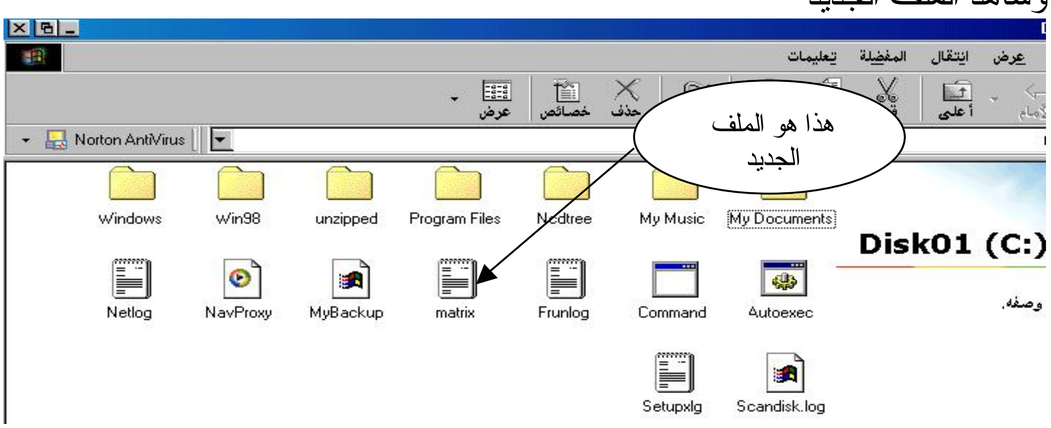
الرمز w لاستحداث نص جديد للكتابة

اسم الملف ونحدد موقعه( مساره) في القرص C

نلاحظ أننا قمنا في السطر الأول أعلاه بإنشاء موقع للملف في ذاكرة الحاسوب طبعا نكتب FILE بالأحرف الأبجدية الكبرى ثم بعد ذلك * وبعدها نطلق المسمى مثلا f

بعد ذلك نقوم بعمل الدالة الخاص بفتح ملف في قرص disk وتعود ( تعطي قيمة ) بمؤشر الملف المعرف له وهو f طبعا.
ثم نقوم بتحديد موقع الملف المراد إنشاءه ويجب التحذير هنا انه يجب عدم كتابة اسم ملف من ملفات النظام system لتجنب عدم حدوث خلل ومشاكل بالجهاز مثلا نكتب كما أعلاه matrix.txt وهو ملف نصي ويجب وضع العلامتين \\ بعد كتابة القرص C ولا يصح وضع علامة واحده \ فقط ، بعد ذلك نكتب الرمز w لاستحداث الملف الجديد ثم بعد ذلك في السطر الأخير نقوم بإغلاقه بالدالة (f) fclose ويجب كتابة هذه الدالة كي يصبح البرنامج صحيح .

الآن قم بتنفيذ البرنامج بعد ذلك لن يظهر لك شي قد تستغرب ما الفائدة الآن اذهب عزيزي إلى القرص C وهو الذي قمت بإنشاء الملف فيه ( المسار أعلاه) وشاهد الملف الجديد



جدول الأنماط ، حسب ما قررته ANSI مؤخرا:

| رمز النمط | وظيفته |
|---|---|
| "r" | لفتح ملف النص للقراءة |
| "w" | لاستحدث ملف نص للكتابة |
| "a" | للإلحاق بملف نص |
| "rb" | لفتح ملف ثنائي القراءة |
| "wb" | لاستحداث ملف ثنائي الكتابة |
| "ab" | للإلحاق بملف ثنائي |
| "r+" | لفتح ملف نص للقراءة أو الكتابة |
| "w+" | لاستحداث نص للقراءة أو الكتابة |
| "a+" | لفتح ملف نص للقراءة أو الكتابة |
| "r+b" | لفتح ملف ثنائي للقراءة أو الكتابة |
| "w+b" | لاستحداث ملف ثنائي للقراءة أو الكتابة |
| "a+b" | لفتح ملف ثنائي للقراءة أو الكتابة |

**الجدول ٢ - ٥**

نلاحظ أن هذا الجدول يمكن استعماله لكل من ملفات النص و الملفات الثنائية.

## دالة الكتابة داخل الملف fprintf ()

مثال ٢:

قم بكتابة النص "welcome to c++" داخل الملف الذي قمنا بإنشائه في المثال السابق (1) وهو ملف matrix ؟

```
#include "stdio.h"
main()
{
FILE *f;
f=fopen("C:\\matrix.txt","w");
fprintf (f,"welcome to c++");


return 0;
}
```

اظفنا هذا السطر الجديد للطباعة بداخل الملف

لاحظ عزيزي القارئ أننا قمنا بوضع الدالة (fprintf) وهي الدالة الخاصة بالكتابة داخل الملفات بإمكانك الرجوع للجدول ١-٥ الآن نفذ البرنامج وبعد تنفيذ ارجع للملف في القرص c وافتحه لتشاهد العبارة welcome to c++ قد كتبة بداخله.

## دالة إغلاق الملف fclose ()

وتعمل عكس الدالة fopen() ، فتغلق الملف الذي فتحناه بالدالة fopen() ، وكل الملفات المطلوبة منك إغلاقها قبل نهاية البرنامج ، وفي حالة عدم قيامنا بإغلاق الملف فان عددا من المشكلات قد تقع ، ومنها ضياع بعض المعطيات ، واتلاف الملف ، ووقوع أخطاء في البرنامج .

مثال٣:

```
#include "stdio.h"
main()
{
FILE *f;
f=fopen("C:\\matrix.txt","w");
fprintf (f,"welcome to c++");
fclose (f)

return 0;
}
```

قمنا بإغلاق الملف في هذا السطر كما تلاحظ

## الدالتان ()putw() getw

تقوم معظم مترجمات ++c باستعمال هاتين الدالتين الإضافيتين لتقوما بعمليتي قراءة وكتابة الأعداد الصحيحة من والي ملفات الأقراص ، وهاتان الدالتان غير معتمدين من قبل معهد المقاييس الوطني الأمريكي ANSI ، وتعمل هاتان الدالتان تماما كالدالتين ()getc و ()putc والفرق الوحيد انهما تتعاملان مع عدد صحيح بدلا من رمزي.

مثال٤:

الدالة التالية تقوم بكتابة ( طباعة) العدد الصحيح 1000 في ملف القرص المشار إليه بمؤشر الملف f :

```
Putw (1000,f);
```

مثال٥:

الدالة التالية تقوم بكتابة قيمة المتغير الصحيح x في ملف القرص المشار إليه بمؤشر الملف f :

```
Putw (x,f);
```

مثال٦:

الدالة التالية تقوم بقراءة عدد صحيح ، من ملف مشار إليه بمؤشر الملف f :

```
#include "stdio.h"
main()
{
FILE *f;
int x;
x=getw (f);
printf ("%d",x);
return 0;
}
```

# النهاية

٢٨ تشرين الأول ٢٠٠٣